ESCUELA SUPERIOR DE INGENIERÍA

INGENIERÍA TÉCNICA EN INFORMÁTICA DE GESTIÓN

Reingeniería de una composición en lenguaje WS-BPEL 2.0: ASTRO

Inmaculada Labrador del Río

31 de julio de 2010

ESCUELA SUPERIOR DE INGENIERÍA

INGENIERO TÉCNICO EN INFORMÁTICA DE GESTIÓN

Reingeniería de una composición en lenguaje WS-BPEL 2.0: ASTRO

- Departamento: Lenguajes y sistemas informáticos
- Director del proyecto: Inmaculada Medina Bulo
- Autor del proyecto: Inmaculada Labrador del Río

Cádiz, 31 de julio de 2010

Fdo: Inmaculada Labrador del Río

Licencia

Este documento ha sido liberado bajo Licencia GFDL 1.3 (GNU Free Documentation License). Se incluyen los términos de la licencia en inglés al final del mismo.

Copyright (c) 2010 Inmaculada Labrador del Río.

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.3 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

Agradecimientos

En especial a un miembro del Grupo de Investigación SPI&FM, Antonio García Domínguez, por su apoyo durante todo el desarrollo del proyecto y por sus minuciosas correcciones que han hecho posible la mejora de esta documentación.

A mi tutora del proyecto, Inmaculada Medina Bulo, por todos las reuniones semanales que convocaba, por brindarme la oportunidad de trabajar en el Grupo de Investigación a través de este Proyecto Fin de Carrera, y por su cooperación en todo momento.

En general a todos los miembros del Grupo de Investigación, los cuales también han sido mis profesores durante toda la carrera de Ingeniería Técnica, por sus magistrales clases, en las cuales sin duda, aprendí mucho y nunca olvidaré.

En especial también, a Juan Antonio Tejero, por su cariño en estos años.

Por supuesto, a mis padres, por su dedicación, apoyo y educación que me han dado durante todo este largo y duro camino y toda mi vida.

Índice general

Índice de figuras

Índice de cuadros

1 Introducción

Este Proyecto Fin de Carrera se introduce en las líneas de investigación del grupo de investigación SPI&FM de la Universidad de Cádiz y en los Servicios Web utilizando el lenguaje WS-BPEL, realizando una reingeniería de una composición en ese lenguaje, ya que se ha adaptado desde una versión antigua de WS-BPEL a la reciente cumpliendo los estándares.

1.1. Motivación

Este proyecto arranca por un ofrecimiento por parte del grupo de investigación SPI&FM (Mejora del proceso software y métodos formales[1]), el cual pertenece al Departamento de Lenguajes y Sistemas Informáticos de la Universidad de Cádiz.

Los miembros del Grupo de Investigación junto a mi tutora del Proyecto, Inmaculada Medina Bulo, me propusieron este proyecto de Fin de Carrera con el fin de entrar en el Grupo, y me gustó mucho la idea, ya que me serviría para aprender conocimientos sobre Servicios Web, algoritmos genéticos, lenguaje WS-BPEL entre otros, lo cual sería bastante interesante.
Un problema que se suele encontrar todo el que trabaja con servicios web y el lenguaje WS-BPEL es que existen muy pocos ejemplos de mediana complejidad disponibles. El Proyecto iría un poco a solventar ese problema. Lo principal sería estudiar el lenguaje WS-BPEL, que después de estudiar varios lenguajes en la carrera no supondría mucho problema. ASTRO, es una reingeniería de una tienda virtual estilo *Amazon*®, en lenguaje BPEL4WS 1.1, adaptada a WS-BPEL 2.0, para posteriormente servir como entrada a dos de las herramientas desarrolladas por el Grupo de Investigación, y estudiar la calidad de casos de prueba y los resultados.

[1] en inglés *Software Process Improvement and Formal Methods*

1.2. Grupo de Investigación SPI&FM

Figura 1.1: Logotipo del grupo de Investigación

El grupo de investigación "Mejora del proceso software y métodos formales" se creó en el año 2004 dentro del Plan Andaluz de Investigación, que se le asigna la referencia TIC-195. Está integrado por un grupo de profesores del Departamento de Lenguajes y Sistemas Informáticos y tiene su sede en la Universidad de Cádiz.

Las líneas de investigación del grupo se centran en los campos de la mejora del proceso software y en los métodos formales, teniendo como objetivo final contribuir a mejorar la calidad del software a través de una mejor comprensión de los factores que tanto a nivel del proceso como de producto influyen en ésta.
Las principales líneas de trabajo son:

- **Proceso Software**: Evaluación y Mejora del Proceso Software, modelado y simulación del proceso de Desarrollo de Software: orientado al servicio y centrado en el usuario.
- **Ingeniería de la Web:** Desarrollo dirigido por modelos. Evaluación de la usabilidad. Simulación y pruebas de composiciones de servicios web. Tecnologías del aprendizaje. Web semántica.
- **Métodos Formales:** Métodos formales aplicados al cálculo de Lambek. Verificación formal de algoritmos de álgebra computacional con ACL2.

La importancia económica que están alcanzando las composiciones de servicios en WS-BPEL obliga a que se preste especial atención a la prueba de este tipo de software. Hasta la fecha se han publicado trabajos relacionados con diversos aspectos de la prueba de composiciones de Servicios Web en WS-BPEL[13], [14], [12], [15], la mayor parte de ellos relacionados con la generación de casos de prueba.
En el grupo de investigación trabaja en estas líneas porque los Servicios Web son cada vez más importantes, y sería interesante disponer de un sistema que les permita medir la calidad de los distintos conjuntos de casos de prueba. Una técnica apropiada para realizar este trabajo, sería el análisis de mutaciones de composiciones. Existen diversos trabajos que usan la técnica de mutaciones aplicados a otros lenguajes con éxito, pero la mayoría de estos trabajos no estudian la calidad de los casos de prueba generados. Por esto, la importancia que le dio el Grupo a estudiar la calidad de los casos de prueba aplicado al lenguaje WS-BPEL.
La idea es ver los errores más comunes o frecuentes que tiene el programador al programar o implementar un proceso de negocio en WS-BPEL.

La técnica de análisis de mutaciones utiliza unos operadores (creado por el grupo) de mutación para generar un conjunto de programas a probar. El grupo de investigación ha desarrollado una herramienta que utiliza esa técnica: GAmera.

El grupo ha desarrollado otra herramienta, que utiliza la técnica de generación de invariantes, llamada Takuan[7].

GAmera

Esta herramienta es un sistema que genera mutantes para composiciones en WS-BPEL[2].

El lenguaje WS-BPEL y los lenguajes relacionados con los Servicios Web están siendo muy utilizados y resultando ser muy útiles en los últimos años. Por este motivo se le debe dar una gran importancia y prestar cierta atención a la prueba de este tipo de software. La prueba de mutaciones es una técnica que ha sido aplicada con éxito a muchos programas de distintos lenguajes. Para aplicar la técnica de mutaciones se necesita de una aplicación que genere esos mutantes, es decir, una aplicación que decida qué mutantes generar y un conjunto de operadores de mutación para llevarla a cabo, es decir, una serie de operadores que hagan los cambios oportunos.

GAmera[3] es un sistema generador automático de mutantes, creado por el grupo de investigación SPI&FM, para composiciones en lenguaje WS-BPEL. Este sistema está basado en algoritmos genéticos e intenta minimizar el número de mutantes generados y ejecutados, independiente del número y tipo de operadores de mutación.

Takuan

Takuan es otras de las herramientas desarrolladas por el grupo de investigación, libre. Sirve para ayudar en la prueba de composiciones de Servicios Web con WS-BPEL. Para ello, Takuan usa la generación dinámica de invariantes. La técnica de generación de invariantes hace uso de dos partes: métodos formales y prueba masiva de software. La técnica en la que se basa Takuan consiste en detectar propiedades que se mantienen en varios puntos de un programa en WS-BPEL. Las utilidades de este sistema son ayudar en la depuración y verificación de una composición observando si las propiedades obtenidas están en consonancia con las especificaciones de la composición.

[2]El lenguaje WS-BPEL y su correspondiente manual se explicarán en un anexo al final de la presente memoria.
[3]GAmera está explicado más a fondo en el anexo B de la presente memoria.

Otra utilidad que tiene Takuan, es evaluar la calidad de los casos de prueba para probar una composición. Para evaluar la calidad solo basta con usarlo de entrada en Takuan y observar qué invariantes genera.

1.3. Objetivos

Uno de los objetivos es el estudio de la herramienta desarrollada por el grupo de investigación SPI&FM, para la generación automática de mutantes para el lenguaje WS-BPEL (GAmera). Después de esto, el objetivo principal de este Proyecto Fin de Carrera ha sido la de realizar la reingeniería de una composición en WS-BPEL 2.0 llamada ASTRO, para más tarde introducirla como entrada en GAmera y estudiar la calidad de los casos de prueba a través del análisis de mutaciones. Para ello, primeramente, habría que aprender el lenguaje WS-BPEL 2.0 y también aprender la versión BPEL4WS 1.1, aunque no todo, donde más tarde se hablará el porqué de la antigua versión.
ASTRO consiste en una tienda virtual similar a *Amazon*®, la cuál a su vez se divide en cuatro grandes partes, por lo tanto en realidad, este proyecto consiste en **4 composiciones** en WS-BPEL:

- ASTROBookCart: composición que envuelve los servicios de un carrito de compra virtual.
- ASTROBookSearch: composición que envuelve los servicios de la búsqueda de un libro en ASTRO.
- ASTROBookBank: composición que envuelve los servicios de un banco.
- ASTROBookStore: envuelve a los tres anteriores en una sola composición.

El principal problema era que ASTRO está en BPEL4WS 1.1 (BPEL for Web Services 1.1) y no sigue el estándar, es una versión antigua la cual no se podía utilizar con la versión de ActiveBPEL que utiliza GAmera, ya que solo trabaja con composiciones WS-BPEL 2.0. Por lo tanto, el Proyecto ha sido una adaptación de la versión antigua 1.1 a la nueva versión WS-BPEL 2.0, es decir, una reinigeniería de una composición que ya existe, en una versión antigua incompatible con estas dos herramientas del Grupo (GAmera y Takuan).

Posteriormente, ASTRO serviría como entrada a GAmera para generar los mutantes de esta composición y poder así estudiar más a fondo esta potente herramienta creada por el Grupo de Investigación, ya que desde el Grupo se necesitaba una composición lo suficientemente grande como para generar los mutantes necesarios para realizar el análisis. También la realización de los suficientes casos de pruebas, tanto en número como en calidad de casos de prueba para matar el mayor número de mutantes posibles.

2 Desarrollo del calendario

2.1. Diagrama de Gantt

El diagrama de Gantt es una herramienta cuyo objetivo es mostrar el tiempo de dedicación previsto para diferentes tareas o actividades a lo largo de este Proyecto Fin de Carrera. Las tareas se representarán mediante bloques, se podrán solapar y no se tiene que terminar completamente la elaboración de una para el comienzo de la siguiente ya que no son actividades críticas [1].
La realización de este Proyecto Final de Carrera ha tenido un coste de elaboración total de 9 meses, abarcando desde noviembre del 2009 hasta finales de julio del 2010, aunque la dedicación a la elaboración de este PFC no ha sido plena, ya se ha tenido que compaginar en diciembre de 2009 con exámenes de esa misma convocatoria, y también con prácticas de empresa en la Universidad de Cádiz.
Gracias al "Diagrama de Gantt" se puede ver gráficamente y de una manera clara el solapamiento del que hablamos. A continuación, se especifican y se numeran las distintas tareas que se han llevado a cabo para la realización de este PFC.

1. En el primer bloque se estudia el lenguaje de programación WS-BPEL 2.0, también se estudia algo de BPEL4WS 1.1. Se miran las especificaciones facilitadas por el tutor. Este período abarca desde noviembre de 2009 y dura hasta febrero de 2010.
2. El segundo periodo abarca desde febrero de 2010 hasta finales de abril de 2010, con el estudio y análisis de las composiciones de ASTRO. En este periodo se adapta también la composición, y el desarrollo de las interfaces WSDL que modelarían la conexión con *Amazon*®.
3. Este tercer periodo abarca desde abril de 2010 hasta finales de julio de 2010, y se trata del desarrollo de los casos de prueba, lo que sería el BPTS (BPEL Test Cases). Este ha sido sin duda, el más duro, debido a la complejidad de las 4 composiciones de ASTRO, sobre todo la composición *ASTROBookStore* que es la que orquesta toda la tienda virtual en una sola. Debido a la gran variedad de ramas y ejecuciones posibles sobre esta composición, ha sido el conjunto de casos de prueba más complicado de realizar.
4. La memoria se empezó a realizar a finales de marzo de 2010, ya que se consideró que hasta entonces no se tenía madurez completa sobre todos los conceptos que componen este Proyecto Fin de Carrera.

[1] Las actividades críticas pueden aparecer en un diagrama de Pert por ejemplo

Cuadro 2.1: Lista de tareas asociadas al Proyecto Fin de Carrera

Nombre	**Inicio**	**Fin**
Estudio, aprendizaje de WS-BPEL	5/11/09	02/10
Análisis composición	25/02/10	25/03/10
Adaptación de ASTROBookCart 1.1 a BPEL 2.0	25/02/10	21/05/10
Adaptación de ASTROBookSearch 1.1 a BPEL 2.0	25/02/10	22/05/10
Adaptación ASTROBookBank 1.1 a BPEL 2.0	25/03/10	25/05/10
Adaptación ASTROBookStore 1.1 a BPEL 2.0	5/04/10	19/06/10
Implementación casos de prueba para ASTROBookCart	5/04/10	26/06/10
Implementación casos de prueba para ASTROBookSearch	8/04/10	29/6/10
Implementación casos de prueba de ASTROBookBank	31/03/10	29/05/10
Implementación casos de prueba de ASTROBookStore	31/04/10	22/07/10
Diseño	4/01/10	4/07/10
Memoria	15/03/10	1/08/2010
Pruebas	1/06/2010	30/07/10

En la tabla 2.1 se pueden visualizar la lista de tareas con los tiempos.

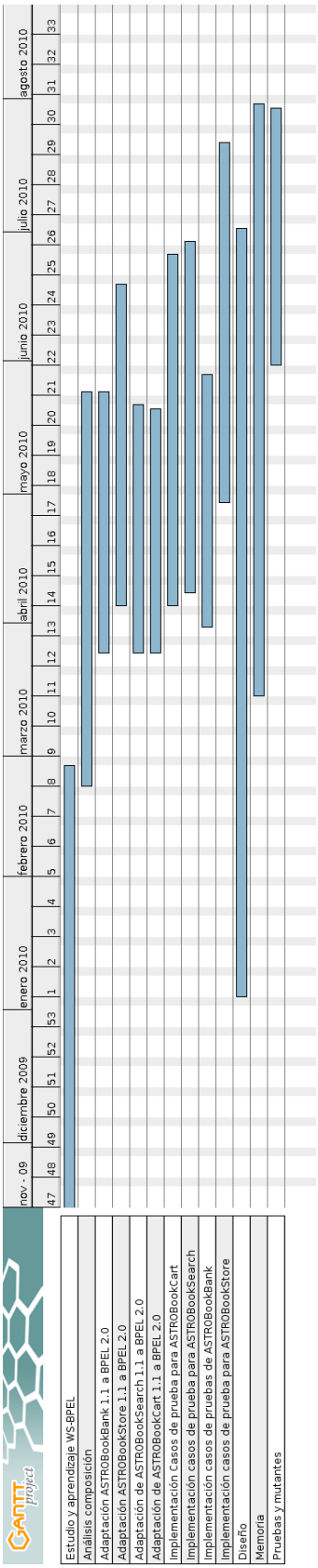


Figura 2.1: Diagrama de Gantt asociado al Proyecto

3 Descripción general del proyecto

3.1. Funciones del producto

ASTRO es una tienda virtual de compra-venta de libros, estilo *Amazon*®. ASTRO se presenta como un proyecto de investigación en el ámbito de los Servicios Web y aplicaciones orientadas a servicios. Su ámbito, a pesar de que WS-BPEL sea un lenguaje comercial, es hacia la investigación ya que el objetivo de la composición es servir para el estudio y la medición de la calidad de los casos de prueba.

Como se ha explicado en el capítulo 1.2, SPI&FM es un grupo cuyas líneas de investigación se centran en los campos de la mejora del proceso software y en los métodos formales, teniendo como objetivo final contribuir a mejorar la calidad del software a través de una mejor comprensión de los factores que tanto a nivel de proceso como de producto influyen en ésta. Para ello, cuentan con GAmera[1], para realizar estas pruebas y mejorar la calidad de los casos de prueba. También cuentan con otra herramienta para ayudar en la prueba de composiciones de Servicios Web con WS-BPEL, usando la generación dinámica de invariantes cuya técnica toma lo mejor de dos mundos: la prueba masiva de software y los métodos formales (llamada Takuan). Ambas, como se habló en 1.2, desarrolladas por el Grupo. Este PFC, por lo tanto, tiene la misión de servir como entrada a ambas herramientas, para su posterior estudio, mediante la generación de mutantes, la calidad de los casos de prueba y ver los errores más comunes cometidos al realizar la reingeniería en ASTRO. En la figura 3.1 se puede ver la lógica que sigue ASTRO, como podemos ver, tenemos las tres funciones principales, que se unen en uno solo.

3.1.1. Reingeniería código ASTRO

El problema que presenta ASTRO es que está implementado en una versión antigua de BPEL, la versión BPEL4WS (BPEL for Web Services) 1.1, por lo que no se puede utilizar ni con GAmera ni con Takuan, debido a que no estaba en WS-BPEL 2.0, y no sigue el estándar.
Por lo tanto, el objetivo de este Proyecto ha sido la de realizar una nueva versión de ASTRO en WS-BPEL 2.0, siguiendo el estándar y pudiéndose utilizar dentro del Grupo de Investigación finalmente.

[1]De GAmera se hablará en el apéndice B a fondo

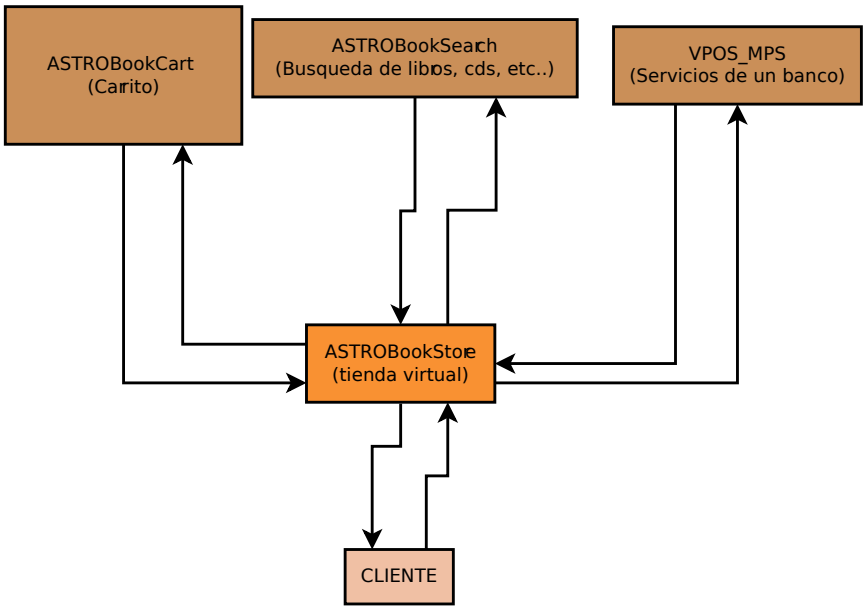


Figura 3.1: Figura del esquema de ASTRO

ASTRO se compone de 4 composiciones WS-BPEL, las cuales son: *ASTROBookSearch*, *ASTROBookCart*, *ASTROBookBank* y *ASTROBookStore*. Describamos con profundidad cada una de las partes.

ASTROBookSearch

Esta es la composición que envuelve los servicios de la búsqueda de la tienda virtual.

Para esta composición, primero se ha adaptado a la nueva versión WS-BPEL 2.0 cumpliendo los estándares. Se ha tenido que adaptar algunos aspectos de la interfaz WSDL, pero no todo, ya que el lenguaje que define la interfaz (WSDL) es en realidad el mismo usado con WS-BPEL 2.0 que con BPEL4WS 1.1, como por ejemplo, en un elemento `<partnerLinkType>`, en la antigua versión tenía como elemento anidado `<portType>`, aunque en la nueva versión 2.0 es un atributo del `<partnerLinkType>`, como vemos a continuación:

```
<plnk:partnerLinkType name="ASTROBookSearch_PLT">
   <plnk:role name="ASTROBookSearch_Customer" portType="
      tns:ASTROBookSearch_CallbackPT"/>
   <plnk:role name="ASTROBookSearch_Service" portType="
      tns:ASTROBookSearch_PT"/>
</plnk:partnerLinkType>
```

Mientras que en la versión 1.1, es un elemento anidado como sigue:

```
  <plnk:partnerLinkType name="ASTROBookSearch_PLT">
   <plnk:role name="ASTROBookSearch_Customer">
      <plnk:portType name="tns:ASTROBookSearch_CallbackPT"/>
   </plnk:role>
   <plnk:role name="ASTROBookSearch_Service">
      <plnk:portType name="tns:ASTROBookSearch_PT"/>
   </plnk:role>
</plnk:partnerLinkType>
```

Una vez adaptada la composición y algunos aspectos de WSDL, se ha procedido al desarrollo de otra interfaz que modelara el agente externo *Amazon*®, puesto que la composición modela el Servicio Web real de conectarse con ella, se ha tenido que implementar. También se ha tenido que realizar el *XML Schema* correspondiente a la estructura de los datos que utiliza el WSDL de *Amazon*®.
Una vez realizada todas las dependencias necesarias para poner en marcha la composición, se ha desarrollo el conjunto de casos de prueba. Para la realización del mismo se ha tenido bastantes problemas, debido a la complejidad del mismo, ya que los mensajes enviados y recibidos son asíncronos y no se sabía con certeza cuando se irían a recibir.
El conjunto de casos de prueba, sirve para modelar esos mensajes recibidos y enviados

entre el cliente, la tienda virtual y los agentes externos.
Esta composición, recibe una petición de *login* de un cliente que desea acceder a la búsqueda de algún libro de la tienda virtual, cuya petición se modela a través del envío de un mensaje con *mockups* (un *mockup* es un fichero con extensión `.bpts` (BPEL Test Suite), el cual se representan los mensajes enviados y recibidos con el cliente; en resumen, un conjunto de casos de prueba para un una composición BPEL determinada), y el proceso BPEL empieza a trabajar con esa petición del cliente. Si hay éxito en el inicio de sesión en la tienda virtual ASTRO, se prosigue con un envío de mensaje al cliente con un *«ok»*, modelado también a través de *mockups*, simulando el intercambio real de mensajes entre el cliente y la tienda virtual ASTRO.
A partir de aquí, se comienza con la búsqueda de libros, mediante una petición del cliente previamente iniciada la sesión. En todo momento se comprueba que todo va bien, mediante control de flujo. Si se produce algún error, se le enviará un mensaje al cliente con el tipo de error. En la figura 3.2 de la página 23 se muestra un esquema del intercambio de mensajes entre el proceso WS-BPEL y el cliente, que en este caso sería el *mockup*(BPTS) que modela el intercambio real de un Servicio Web.

ASTROBookCart

Esta parte envuelve los servicios de un carrito virtual cualquiera en la tienda virtual.

Se ha procedido a su desarrollo, igual que para *ASTROBookSearch*. Esta composición comienza con la petición de un mensaje del cliente para la creación de un carrito e introducir libros en él (previamente buscados en la anterior composición *ASTROBookSearch*). Posteriormente, se intenta "conectar" realmente con Amazon®, pero se ha simulado también con la realización de una interfaz para ello (la realización de un WSDL que contiene las especificaciones de los mensajes que se realizarían si conectase realmente con *Amazon*®), y *mockups*.

Una vez que se ha creado el carrito correctamente, se comienza a introducir libros en el carrito virtual, mediante el intercambio de mensajes entre la simulación de mensajes entre *Amazon*® y ASTRO.
Si no es el caso, se envía un mensaje al cliente con el correspondiente error. En la figura 3.3 de la página 24 se muestra un esquema de intercambios de mensajes de esta composición.

ASTROBookBank

Esta parte envuelve los servicios de un banco.
Se ha procedido a su desarrollo, igual que para *ASTROBookSearch*, con la diferencia que en esta composición no se modela la interacción real con *Amazon*®, y no se ha realizado WSDL para ello.

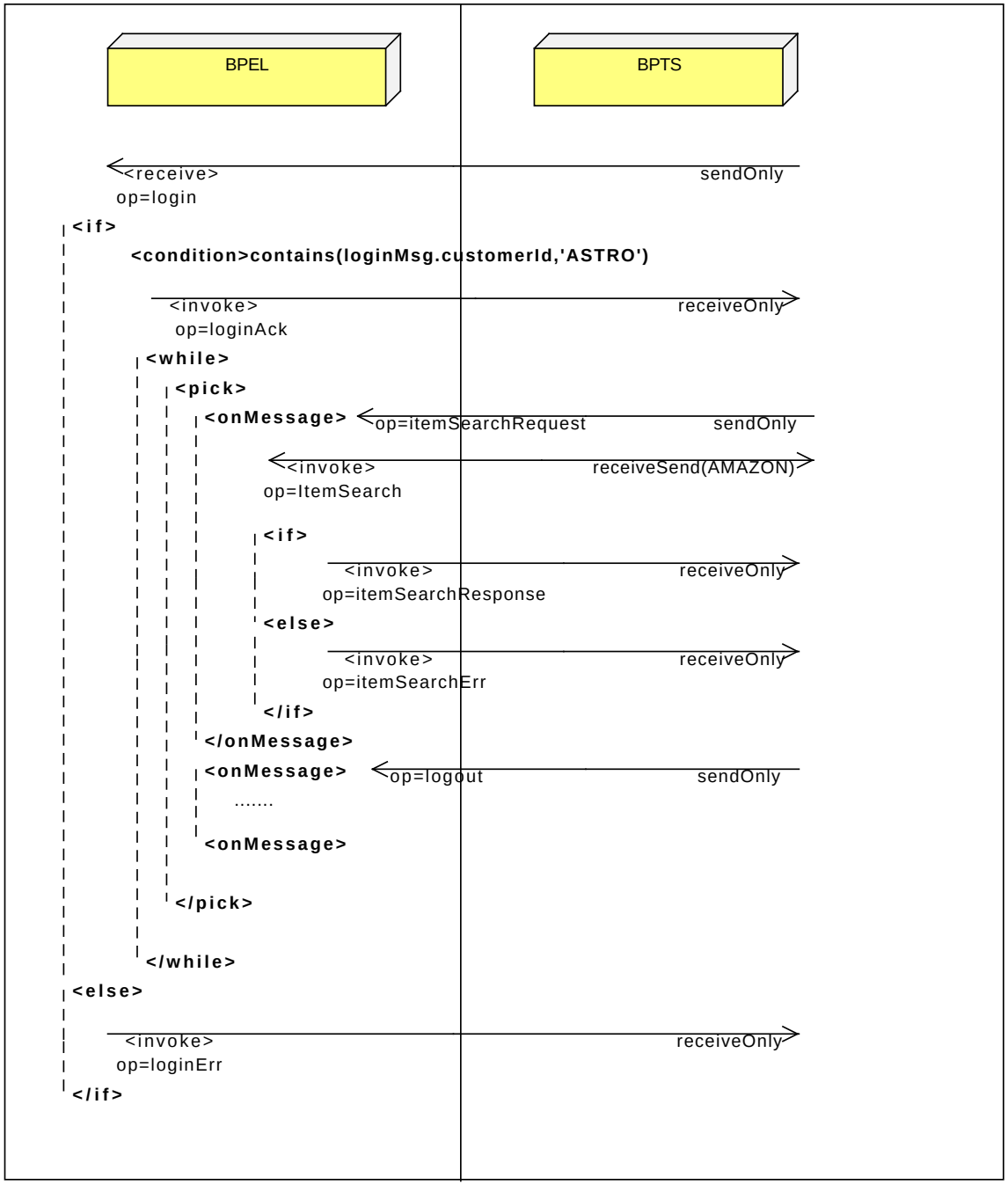


Figura 3.2: Esquema de intercambio de mensajes de *ASTROBookSearch*

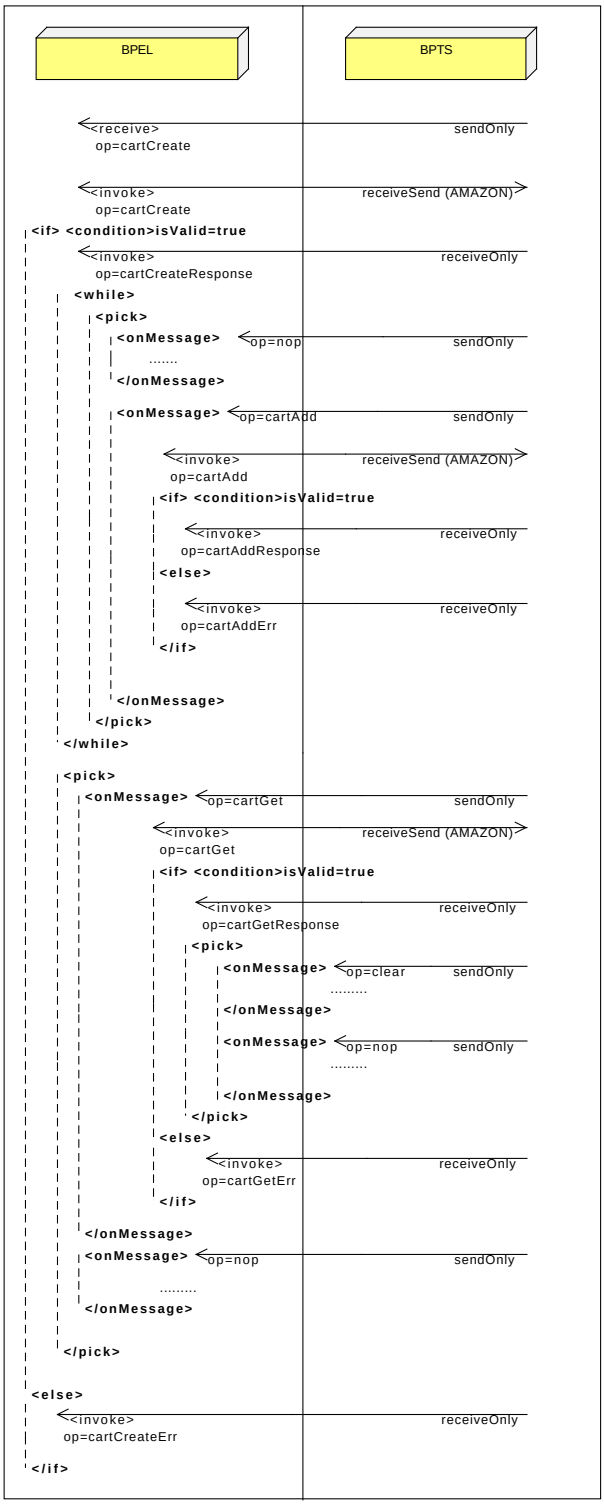


Figura 3.3: Esquema de intercambio de mensajes de *ASTROBookCart*

Esta composición comienza con la petición del cliente de realizar la transacción (previamente con el carrito virtual creado en la anterior composición *ASTROBookCart*). Si todos los datos que se reciben son correctos, se procede a realizar la transacción simulando la conexión con una sucursal bancaria real, modelado mediante *mockups*.
Si en algún momento falla la transacción, se envía un mensaje con el correspondiente error.
En la figura 3.4 de la página 26 se muestra el esquema del intercambio de mensajes.

He de decir que el desplegado y la posterior ejecución de estas tres composiciones son independientes entre sí. Es decir, cada una tiene su conjunto de casos de prueba, y cada composición se ejecuta independientemente del resultado de las demás.

ASTROBookStore

En esta última composición se realiza la orquestación de las tres anteriores composiciones en una sola. *ASTROBookStore* tiene su propio conjunto de casos de prueba.

Esta composición es la más laboriosa sin duda, ya que une las 3 anteriores en una sola, con gran variedad de casos que pueden darse. Las figuras 3.5 de la página 27, 3.6 de la página 28, 3.7 de la página 29 y 3.8 de la página 30 muestran el esquema de intercambio de mensajes. Como se puede observar es bastante largo, con lo cual es bastante complejo realizar el conjunto de casos de prueba para esta composición.

3.1.2. Parte de investigación

Una vez realizada la reingeniería de las 4 composiciones de las que se compone ASTRO, se ha procedido al estudio de la calidad de los casos de prueba. Las 4 composiciones sirven de entrada tanto a GAmera como a Takuan.
El objetivo en la parte de investigación es mejorar la calidad del conjunto de casos de prueba, donde interviene GAmera, haciendo uso de *mutationtool*. La mejora de la calidad de los casos de prueba consiste en "matar mutantes", y cuanto mayor sea el número de mutantes muertos, mejor calidad tendrán los casos de prueba. Esta mejora de calidad se realizará en los siguientes pasos:

1. Generación de mutantes con *mutationtool*. Se generan los mutantes correspondientes a cada composición.
2. Comparación de mutantes. Se comparan los mutantes con la composición original. Se generará un fichero por cada operador de mutación, donde en su interior habrá una matriz M_{xy}, donde x es el número de mutantes diferentes para ese operador en cuestión, y sería el número de casos de prueba que tenemos para esa composición que estamos analizando. Esta matriz está compuesta por 0 y 1, dependiendo de si para un caso de prueba en particular(y) mata a un mutante

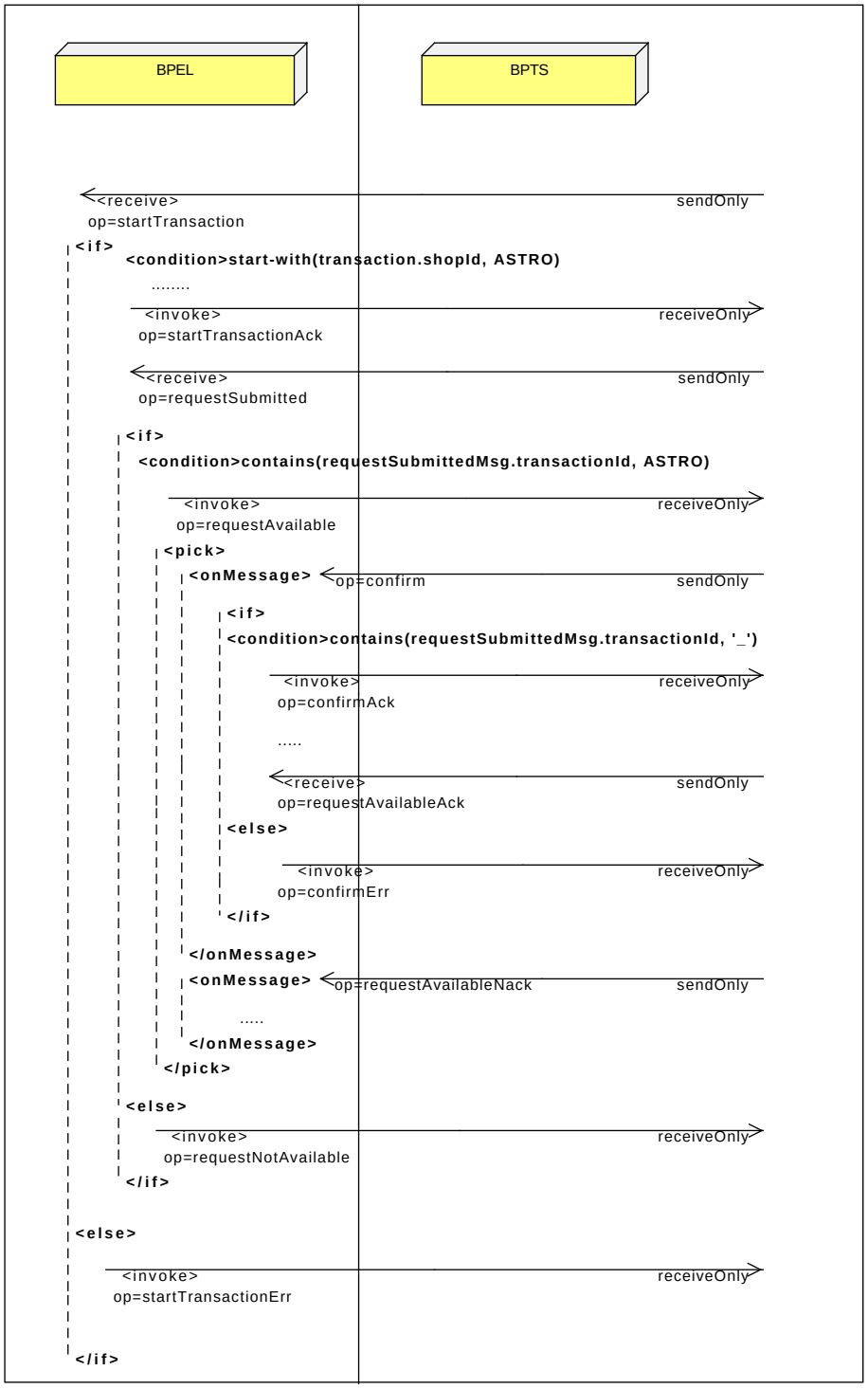


Figura 3.4: Esquema de intercambio de mensajes de *ASTROBookBank*

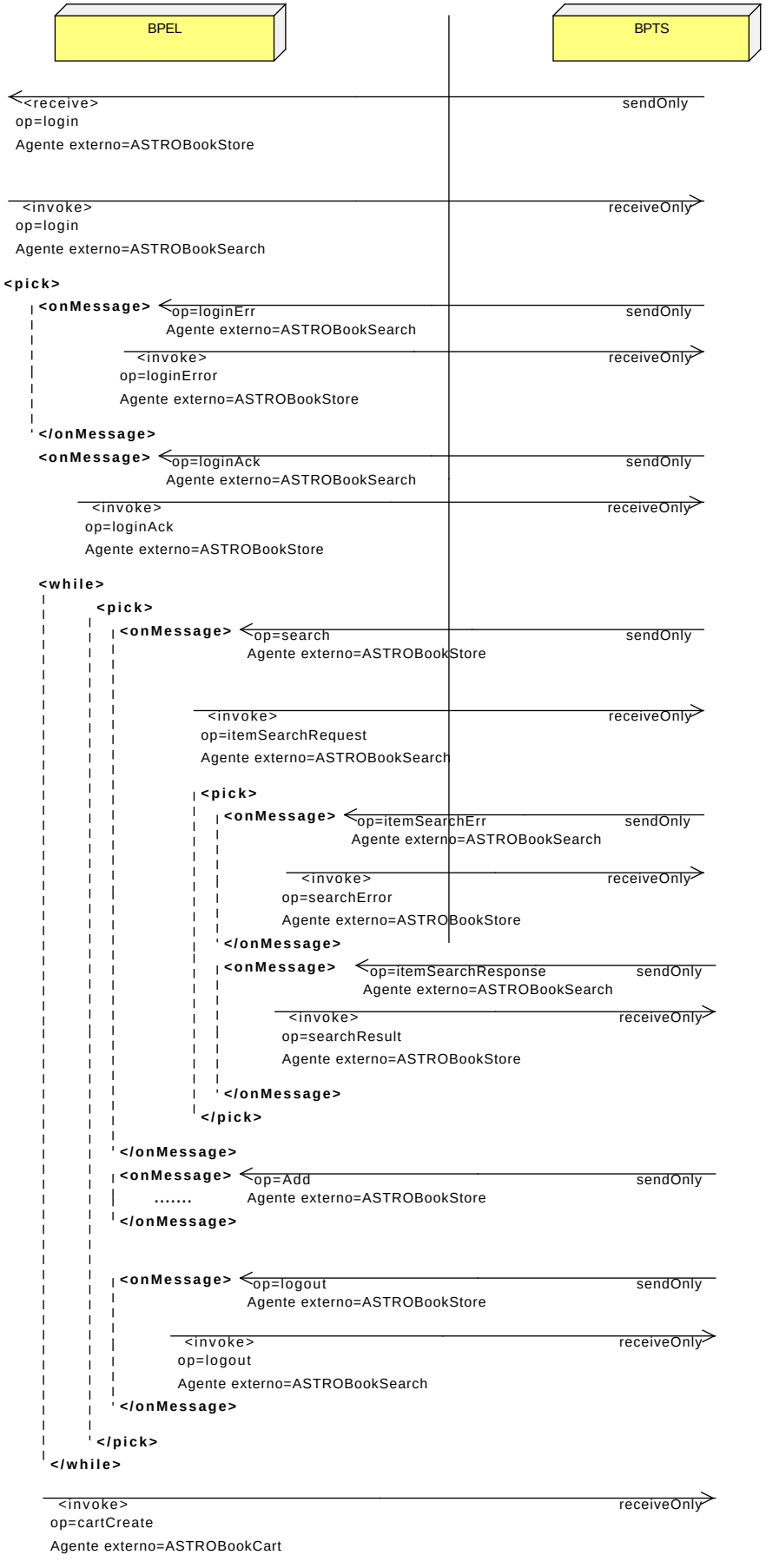


Figura 3.5: Esquema de intercambio de mensajes de *ASTROBookStore*. Parte 1

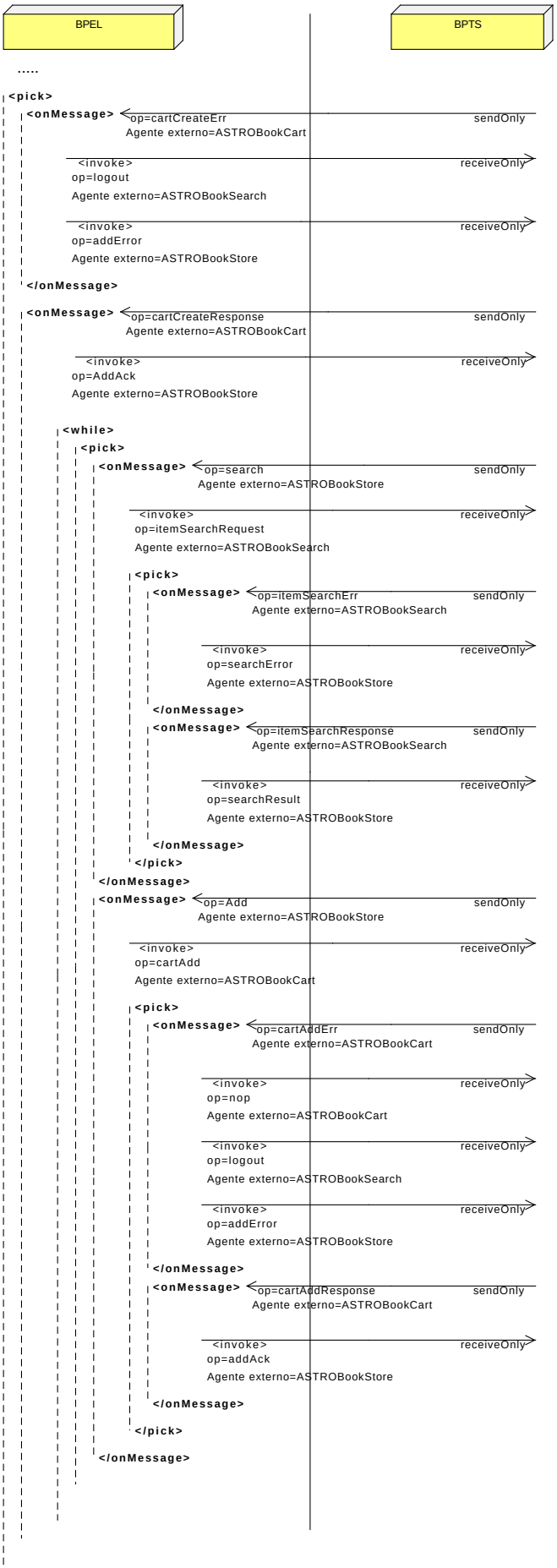


Figura 3.6: Esquema de intercambio de mensajes de *ASTROBookStore*. Parte 2

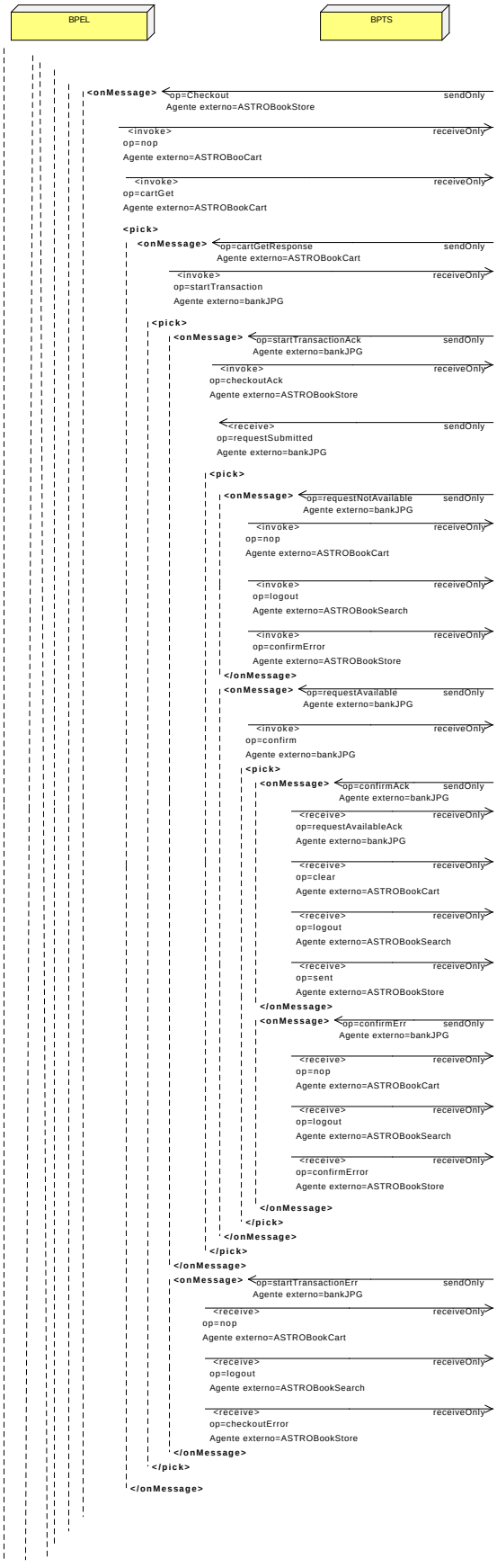


Figura 3.7: Esquema de intercambio de mensajes de *ASTROBookStore*. Parte 3

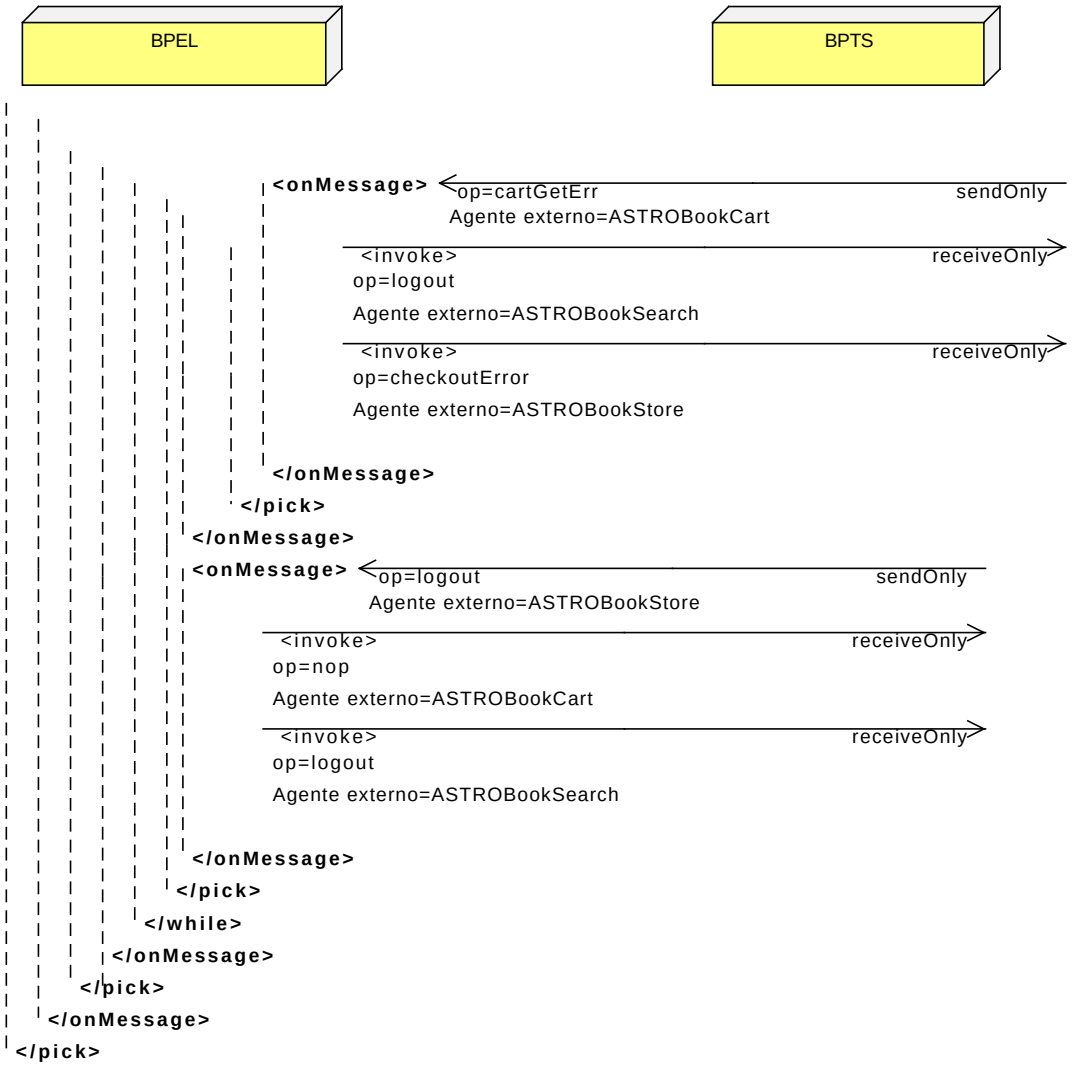


Figura 3.8: Esquema de intercambio de mensajes de *ASTROBookStore*. Parte 4

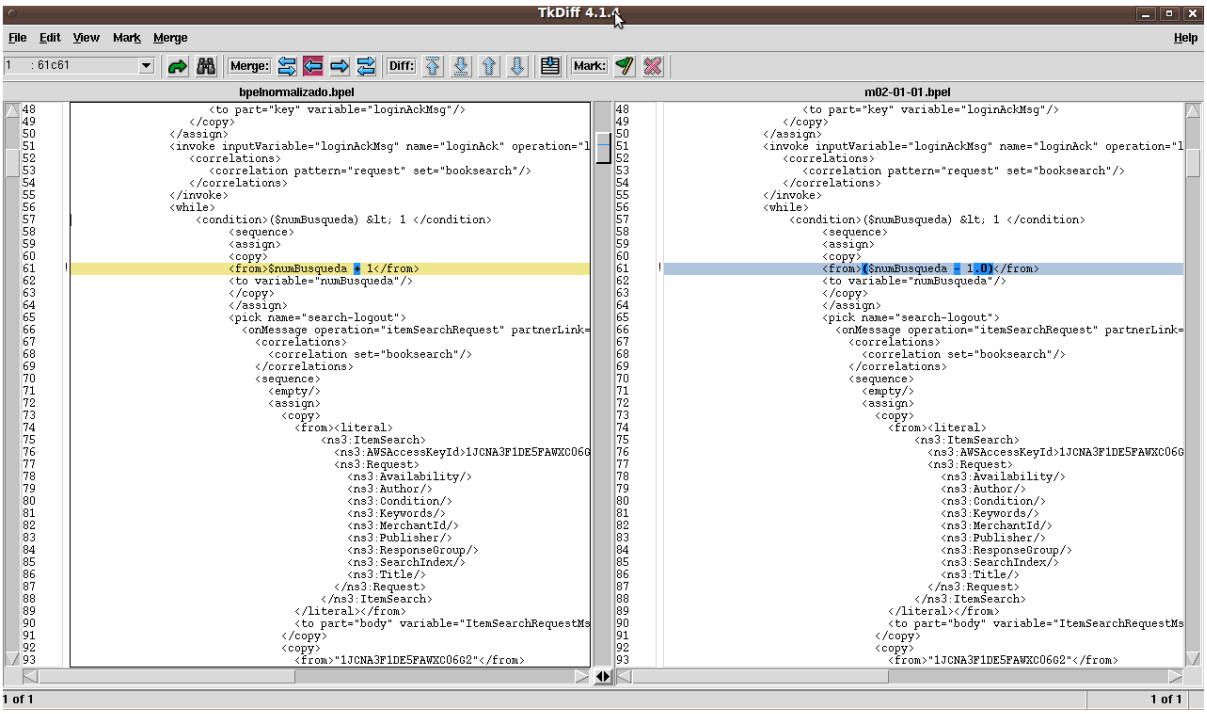


Figura 3.9: Comparación de una composición original y su mutación

(x) o sigue vivo. Entonces, si una sola fila está llena de ceros, podremos decir que el mutante sigue vivo y ese caso de prueba en particular, no habrá matado al mutante. En cambio, si aparece, a lo sumo un 1, diremos que ese caso de prueba habrá matado al mutante. Es decir:
$M_{xy} = 1$ ó $M_{xy} = 0$

3. Matar mutantes vivos para mejorar la calidad del caso de prueba. Para localizar al mutante vivo, debemos hacer uso de la herramienta libre TKDIFF, cuya herramienta muestra a pantalla completa (mitad y mitad), la composición mutada y la original, pudiendo ver dónde está en concreto ese cambio realizado, como se puede ver en el ejemplo 3.9.

En la figura podemos observar el cambio realizado y analizar el porqué el resultado del caso de prueba para el mutado es el mismo que para la composición original, y poder mejorar ese caso de prueba en particular.

Este resultado puede deberse a una mala implementación de la composición, o no hay un caso de prueba que ejecute una rama y las instrucciones de esa rama en particular. Los resultados de ejecutar estas composiciones en GAmera, están en la sección 4.5 del capítulo 4.

3.2. Lenguajes de programación

Los lenguajes utilizados en la realización de este PFC son varios.

3.2.1. Lenguaje WS-BPEL

WS-BPEL es un lenguaje basado en XML que permite especificar el comportamiento de un proceso de negocio basado en interacciones con Servicios Web (WS).
Este lenguaje ha sido utilizado para realizar la reingeniería del proceso de negocio ASTRO, ya que esta composición está en BPEL4WS 1.1 y se ha tenido que adaptar a WS-BPEL 2.0, como se ha dicho anteriormente.
En el apéndice A se explica a fondo sobre este lenguaje.

3.2.2. WSDL 1.1

WSDL[2] son las siglas de Web Services Description Language, un formato XML que se utiliza para describir servicios Web. La versión 1.0 fue la primera recomendación por parte del W3C y la versión 1.1 no alcanzó nunca tal estatus. La versión 2.0 se convirtió en la recomendación actual por parte de dicha entidad.
WSDL describe la interfaz pública a los servicios Web. Está basado en XML y describe la forma de comunicación, es decir, los requisitos del protocolo y los formatos de los mensajes necesarios para interactuar con los servicios listados en su catálogo. Las operaciones y mensajes que soporta se describen en abstracto y se ligan después al protocolo concreto de red y al formato del mensaje.
Se ha combinado WSDL con SOAP y XML Schema. Un programa cliente que se conecta a un Servicio Web puede leer el WSDL para determinar qué funciones están disponibles en el servidor. Los tipos de datos especiales se incluyen en el archivo WSDL en forma de XML Schema. El cliente puede usar SOAP para hacer la llamada a una de las funciones listadas en el WSDL.

3.2.3. XML Schema 1.0

XML Schema es un lenguaje de esquema utilizado para describir la estructura y las restricciones de los contenidos de los documentos XML de una forma muy precisa, más allá de las normas sintácticas impuestas por el propio lenguaje XML. Se consigue así una percepción del tipo de documento con un nivel alto de abstracción. Fue desarrollado por el World Wide Web Consortium (W3C).

[2]Se describe también WSDL en el apéndice A

XML Schema es un lenguaje de esquema escrito en XML, basado en la gramática y pensado para proporcionar una mayor potencia expresiva que las DTD, menos capaces al describir los documentos a nivel formal.

Los documentos esquema (su extensión es `.xsd` de XML Schema Definition (XSD))busca nuevas capacidades a la hora de definir estructuras para documentos XML. El principal aporte de XML Schema es el gran número de tipos de datos que incorpora. De esta manera, XML Schema aumenta las posibilidades y funcionalidades de aplicaciones de procesado de datos, incluyendo tipos de datos complejos como fechas, números y strings.

3.2.4. SOAP

SOAP (siglas de Simple Object Access Protocol) es un protocolo estándar que define cómo dos objetos en diferentes procesos pueden comunicarse por medio de intercambio de datos XML. Este protocolo deriva de un protocolo creado por David Winer en 1998, llamado XML-RPC. SOAP fue creado por Microsoft, IBM y otros y está actualmente bajo el auspicio de la W3C. Es uno de los protocolos utilizados en los servicios Web.
Se habla más a fondo en la sección A.8 del apéndice A.

3.2.5. Lenguaje LaTeX

Para la documentación y elaboración de la memoria del Proyecto se ha utilizado el lenguaje de marcado LaTeX.
LaTeXes un sistema de composición de textos, orientado especialmente a la creación de libros, documentos científicos y técnicos que contengan fórmulas matemáticas. LaTeXestá formado por un gran conjunto de macros de TeX, escrito por Leslie Lamport en 1984, con la intención de facilitar el uso del lenguaje de composición tipográfica, TeX, creado por Donald Knuth. Es muy utilizado para la composición de artículos académicos, tesis y libros técnicos, dado que la calidad tipográfica de los documentos realizados con LaTeX es comparable a la de una editorial científica de primera línea. LaTeX es software libre bajo licencia LPPL.

LaTeXes un sistema de composición de textos que está formado mayoritariamente por órdenes (macros) construidas a partir de comandos de TeX (lenguaje de bajo nivel. Esto es lo que convierte a LaTeXen una herramienta práctica y útil pues, a su facilidad de uso, se une toda la potencia de TeX. Estas características hicieron que LaTeXse extendiese rápidamente entre un amplio sector científico y técnico, hasta el punto de convertirse en uso obligado en comunicaciones y congresos, y requerido por determinadas revistas a la hora de entregar artículos académicos.

LaTeXpresupone una filosofía de trabajo diferente a la de los procesadores de texto habituales (conocidos como WYSIWYG, es decir, *lo que ves es lo que obtienes* y se basa en comandos. Tradicionalmente, este aspecto se ha considerado una desventaja (probablemente la única). Sin embargo, LaTeX, a diferencia de los procesadores de texto de tipo WYSIWYG, permite a quien escribe un documento centrarse exclusivamente en el contenido, sin tener que preocuparse de los detalles del formato. Además de sus capacidades gráficas para representar ecuaciones, fórmulas complicadas, notación científica e incluso musical, permite estructurar fácilmente el documento (con capítulos, secciones, notas, bibliografía, índices analíticos, etc.), lo cual brinda comodidad y lo hace útil para artículos académicos y libros técnicos.
Para instalar LaTeXsimplemente escribimos el siguiente comando por consola:

```
1   sudo apt-get install texlive-full
```

[21]

3.3. Herramientas y tecnologías utilizadas

Para la elaboración de este PFC se han utilizado varias herramientas, entre otras BPELUnit, ActiveBPEL, GAmera, XMLEye, etc.

3.3.1. BPELUnit (Casos de prueba para WS-BPEL)

Las composiciones en BPEL son complejas, por lo tanto, los programas de interacción deben probarse y realizarse pruebas de unidad de caja blanca, al igual que cualquier otro programa de software y asegurar la calidad del mismo. BPELUnit solventa ese problema.
BPELUnit es un framework el cual nos permite testear composiciones en WS-BPEL sin necesidad de utilizar Servicios Web reales, a través de casos de pruebas con *mockups*. Recordando lo que se dijo anteriormente, donde se decía que un **mockup** es un fichero con extensión `.bpts`, en el cual en su interior se representan los mensajes enviados y recibidos con el cliente; en resumen un conjunto de casos de prueba para una composición BPEL determinada.
En el fichero BPTS, se especifican qué datos serán enviados y qué datos serán esperados para cada cliente y compañero (solo queda esperar que el framework de testeo, BPELUnit, haga el resto). Cada caso de prueba contiene un "thread"de cada compañero, llamado *partner track*, el cual contiene una secuencia de interacciones, llamadas actividades, las cuales describen acciones esperadas de *PUT* o acciones que el compañero debe tomar.
A continuación se muestra un ejemplo de un caso de prueba, realizado a la composición de ASTRO:

```
<tes:testCases>
<tes:testCase name="TestCase 1 - Main else- Error en la transaccion"
    basedOn="" abstract="false" vary="false">
  <tes:clientTrack>
        <tes:sendOnly service="ns0:bankJPG_PLTService"
         port="bankJPG_PLTServicePort"
         operation="startTransaction">
                <tes:data>
                        ....
                </tes:data>
        </tes:sendOnly>
  </tes:clientTrack>

 <tes:partnerTrack name="bankJPG_PLT">
        <tes:receiveOnly
                service="ns0:bankJPG_CallbackPLTService"
                port="bankJPG_CallbackPLTServicePort"
                operation="startTransactionErr">
        </tes:receiveOnly>
 </tes:partnerTrack>
</tes:testCase>
```

Se especifican dos compañeros, uno para el cliente y otro para un compañero llamado *bankJPG_PLT*. Cada partner track contiene una secuencia de actividades, según se reciban mensajes de un servicio o se envíen datos. En este caso, sería una actividad asíncrona, ya la actividad está esperando una respuesta asíncrona, para más tarde verificar la respuesta acorde a la condición dada.

BPEL Test Suite (BPTS)

Una buena especificación donde estudiar muy a fondo acerca de como construir un caso de prueba para una composición BPEL, la podemos encontrar aquí: [11].
BPTS es una extensión de archivo, donde se implementa el conjunto de casos de prueba para una composición en WS-BPEL 2.0, y probar la composición. En un fichero BPTS, para probar una composición, funciona del siguiente modo: Una interacción con el *PUT* se le llama actividad. Las secuencias de actividades se denominan *track*: El framework ejecuta un *track* para cada *partner*(compañero) (y uno para el cliente). Los *track* se ejecutan en paralelo y son independientes entre sí. La figura 3.10 muestra un diagrama de secuencia de una interacción típica entre el cliente y el *partner Track* del framework. El PUT es un proceso BPEL sencillo que habla con dos socios para lograr su objetivo: Uno de los socios se invoca de forma asincrónica, y el otro de forma sincrónica. El propio PUT ofrece un funcionamiento sincrónico, que ha sido comprobado por el seguimiento de los clientes. El *partner tracks* y el *client track* prueba los datos de entrada.

	Operación una dirección	Síncrona/asíncrona en dos direcciones
Cliente	El mensaje es enviado por el framework(en este caso BPELUnit) y recibido por el proceso BPEL	Primero el mensaje es enviado por el framework y recibido por el proceso BPEL. El segundo mensaje es enviado por el proceso BPEL, recibido por el framework.
Socio	El mensaje es enviado por el proceso BPEL, recibido por el framework.	El primer mensaje es enviado por el proceso BPEL,recibido por el framework. El segundo mensaje es enviado por el framework, recibido por el proceso BPEL.

Cuadro 3.1: Flujo de mensajes en un caso de prueba

Las actividades de BPELUnit se corresponden con las operaciones y puertos en el WSDL. Cada actividad se corresponde con al menos una operación WSDL:

- **Actividades en una sola dirección y síncronas en dos direcciones.** Estas actividades corresponden exactamente a una operación WSDL dentro de un determinado puerto del WSDL. Una actividad de un solo sentido corresponde a una operación de un solo sentido, lo que significa que la operación WSDL sólo contiene un mensaje. Una actividad sincrónica de dos vías corresponde a una operación de petición-respuesta, lo que significa que la operación WSDL contiene al menos dos mensajes (uno para la entrada, una para la salida, y los fallos adicionales).
- **Actividades asíncronas de dos direcciones.** Las llamadas asincrónicas de dos direcciones se implementan en BPEL mediante dos operaciones WSDL: Una operación que se invoca al principio de la llamada, y otra se utiliza para la devolución de llamada. Las actividades asíncronas en ambos sentidos corresponden, por lo tanto, a dos operaciones WSDL de un solo sentido, que generalmente pertenecen a varios tipos de puerto diferente y por lo tanto los puertos y servicios.

Los flujos de mensajes de un mensaje síncrono y asíncrono puede verse en la tabla 3.1.

Para tener una mejor visión de lo que sería la especificación un conjunto de casos de pruebas, la figura 3.11 muestra la especificación gráficamente de un conjunto de casos de prueba con BPELUnit.

3.3.2. ActiveBPEL 4.1

ActiveBPEL es un motor que nos permite ejecutar los procesos WS-BPEL y con el que BPELUnit puede trabajar a la hora de sustituir los servicios externos por otros emulados (mockups). Es de la compañía ActiveVOS y tiene licencia GPL.
El grupo de Investigación lo ha extendido en varios aspectos para cubrir sus necesidades.

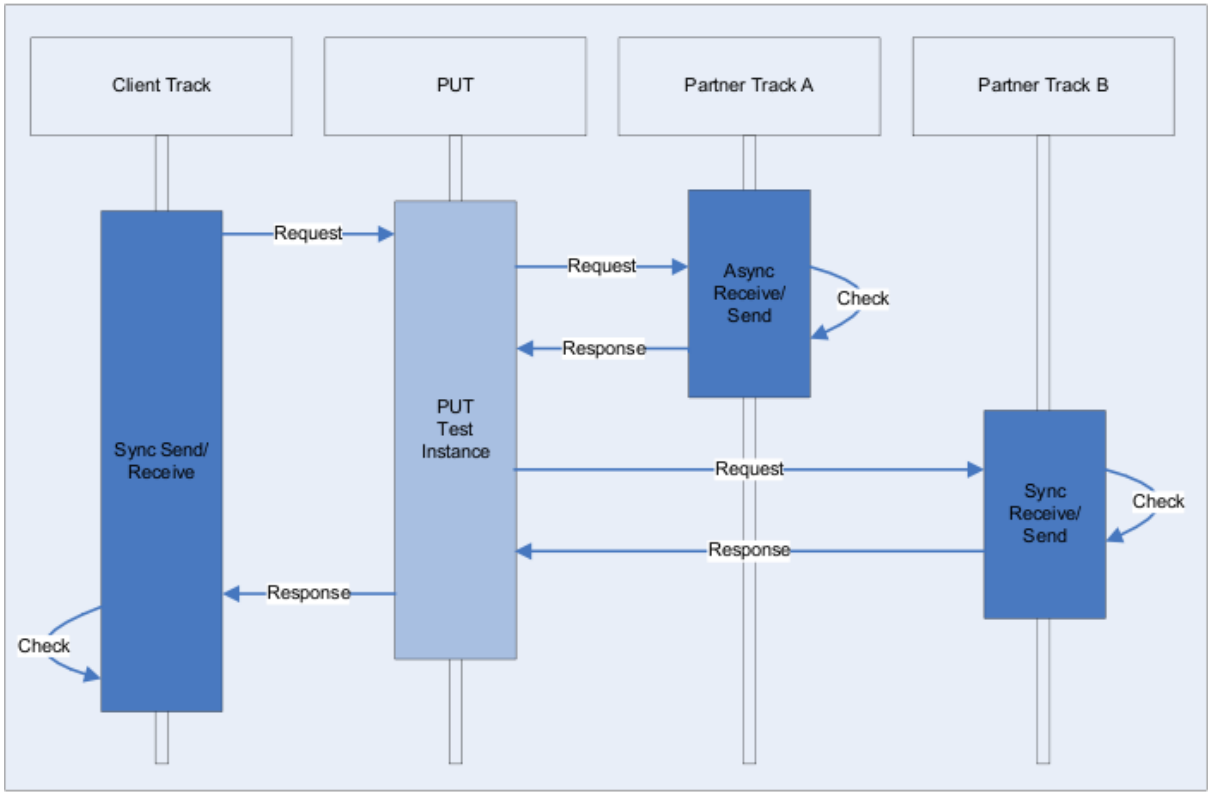


Figura 3.10: Una secuencia de un caso de prueba en BPEL

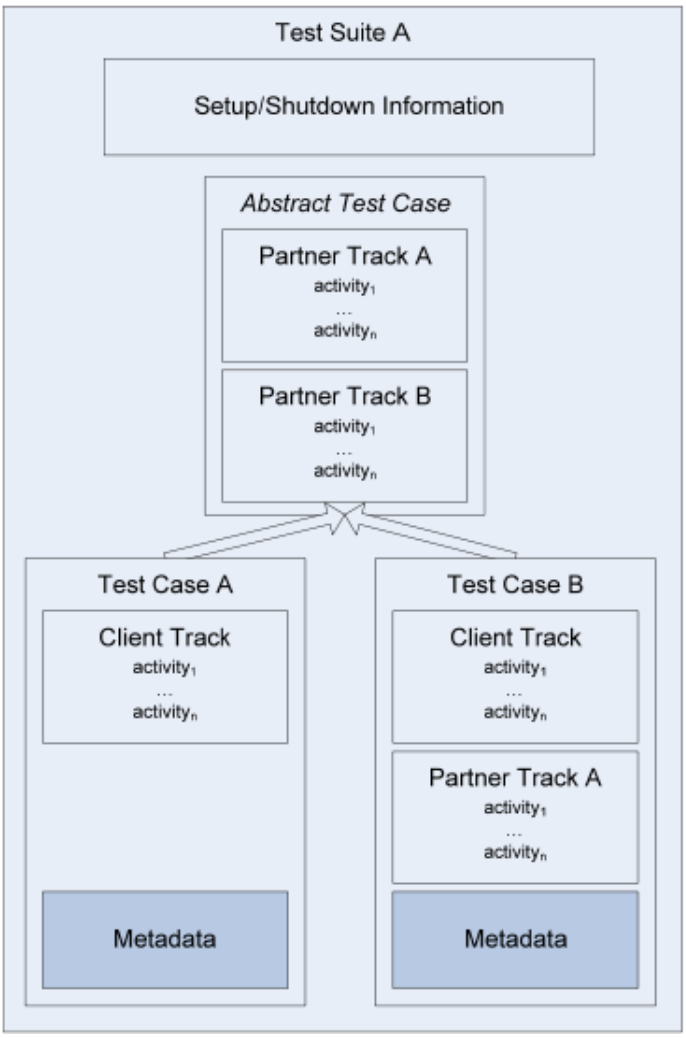


Figura 3.11: Especificación de un conjunto de casos de prueba de BPELUnit

3.3.3. mutationtool

Es la herramienta propia del grupo de investigación para ejecutar las composiciones BPEL y realizar el desplegado, entre otras.
Las funciones de *mutationtool*, son las siguientes:

```
analyze bpel
apply bpel operator operand attribute
applyall bpel
run bpts bpel
compare bpts bpel xml bpel1...
comparefull bpts bpel xml bpel1...
compareout xml1 xml2
normalize bpel
```

Se hablará más a fondo en el manual de usuario de la presente memoria.

3.3.4. XMLEye: transformador y visor genérico de documentos estructurados

XMLEye es una herramienta que fue creada por un miembro del Grupo de Investigación, Antonio García Dominguez para su Proyecto Fin de Carrera en la titulación de Ingeniero Informático en 2008.
Es genérico porque puede abrir sobre todo cualquier tipo de documento. No está atado a ningún formato en particular. Sin embargo, se basa en XML porque todos los documentos de entrada deben ser primero convertidos a XML antes de su visualización.
XMLEye tiene las siguientes funciones:

- Navegación jerárquica simultánea por múltiples documentos a distintos niveles de detalle, con vínculos entre sus elementos y con elementos de otros documentos.
- Búsqueda a lo largo del documento, con diversas opciones de filtrado.
- Integración con nuevos formatos especificados por el usuario, sin necesidad de modificar el código. Cada formato puede especificar el editor y (opcionalmente) el conversor a XML con el que deberá integrarse XMLEye.

  Los formatos son localizables a distintos idiomas, y personalizables por cada usuario del sistema. Se deben de poder restaurar los ajustes originales del formato en cualquier momento.
- Cambio en tiempo de ejecución del comportamiento de visualización del documento y sus nodos, que debe ser extensible por el usuario sin necesidad de añadir código específico a XMLEye.

- Monitorización en segundo plano del documento abierto, volviéndolo a abrir cuando se produzcan cambios en él. Esto posibilita su edición y visualización en paralelo.

Para mi PFC ha sido bastante útil XMLEye, ya que con esta herramienta se puede ver la salida que generan cada uno de los casos de pruebas para una composición WS-BPEL 2.0 en particular, ya que el fichero de salida es `XML`. De manera que si se produce algún error en el envío de mensajes, XMLEye lo detecta, y además aparece el error que se ha cometido y qué mensaje ha fallado. En el manual de usuario se trata a fondo de como utilizar esta herramienta.

3.3.5. TKDIFF

Tkdiff es una interfaz gráfica para comparar diferencias entre dos archivos. Tkdiff se ha utilizado para visualizar la diferencia entre nuestro proceso BPEL original con el mutante que ha generado un operador determinado, puesto que es una comparación línea a línea.
Para visualizar dos archivos, solo tenemos que introducir la siguiente orden por pantalla:

```
1 tkdiff bpeloriginal.bpel mutante.bpel
```

3.3.6. Control de versiones con Subversion (SVN)

Subversion es un software de sistema de control de versiones[2]. Es software libre bajo una licencia de tipo Apache/BSD y se le conoce también como *svn* por ser ese el nombre de la herramienta utilizada en la línea de órdenes.
El repositorio es como un servidor de ficheros ordinario, excepto porque recuerda todos los cambios hechos a sus ficheros y directorios. Esto le permite recuperar versiones antiguas de sus datos, o examinar el historial de cambios de los mismos.

Subversion puede acceder al repositorio a través de redes, lo que le permite ser usado por personas que se encuentran en distintos ordenadores. A cierto nivel, la posibilidad de que varias personas puedan modificar y administrar el mismo conjunto de datos desde sus respectivas ubicaciones fomenta la colaboración. Se puede progresar más rápidamente sin un único conducto por el cual deban pasar todas las modificaciones. Y puesto que el trabajo se encuentra bajo el control de versiones, no hay razón para temer por que la calidad del mismo vaya a verse afectada, si se ha hecho un cambio incorrecto a los datos, simplemente deshaga ese cambio.

Con SVN se sigue la historia de los archivos y directorios a través de copias y renombrados, las modificaciones (incluyendo cambios a varios archivos) son atómicas,

la creación de ramas y etiquetas es una operación más eficiente, se envían sólo las diferencias en ambas direcciones, puede ser servido mediante Apache, sobre WebDAV/-DeltaV. Esto permite que clientes WebDAV utilicen Subversion en forma transparente. Maneja eficientemente archivos binarios, permite selectivamente el bloqueo de archivos. Se usa en archivos binarios que, al no poder fusionarse fácilmente, conviene que no sean editados por más de una persona a la vez.

La estructura habitual de un repositorio de Subversion es:

- Trunk: desarrollo principal.
- Tags: ubicación de las versiones congeladas.
- Branches: ubicación con versiones de desarrollo paralelas al trunk.

3.4. Características de los usuarios

Como ya hemos comentado, ASTRO es un proyecto orientado a la investigación, luego los usuarios de esta aplicación serán aquellos que necesiten trabajar con algoritmos mutantes en el lenguaje BPEL, desarrolladores de composiciones en WS-BPEL o miembros propios del Grupo de Investigación.

3.5. Suposiciones y dependencias

La principal dependencia que afecta a ASTRO viene dada por GAmera. GAmera tiene que ejecutarse bajo un entorno Sun Java 5. Casi todo funciona con OpenJDK 6, pero al ejecutar GAmera se producen unos fallos que no ocurren al usar Sun Java 5, por lo que se optó por dejarlo de ese modo. También ASTRO se ve afectada por:

- Tomcat 5.5
- ActiBPEL 4.1
- BPELUnit
- Gestor de mutantes WS-BPEL

Para ejecutar ASTRO o cualquier proceso de negocio en WS-BPEL 2.0 debe instalarse GAmera en la máquina donde se vaya a ejecutar la composición[3]

[3]Más detalle de instalación y manual de usuario en el capítulo 7 y 8 de la presente memoria.

4 Desarrollo del proyecto

En la realización de un proyecto software se necesita en todo momento de una metodología de desarrollo para su elaboración, especificación de requisitos, etc. Una metodología de desarrollo es un conjunto de pasos, procedimientos, técnicas, herramientas y un soporte documental que posibilita el desarrollo sistemático de software.

Para realizar el desarrollo del proyecto, existe METRICA V3 que ofrece un instrumento útil para la sistematización de las actividades y que dan soporte al ciclo de vida del software.

La metodología METRICA versión 3 cubre los distintos tipos de desarrollo: estructurado y orientado a objeto y es independiente del lenguaje de programación utilizado en la implementación del sistema. Como buen sistema, deberá cumplir todas las normas de Calidad propuestas. Como bien documenta y explica la METRICA V3, existen una serie de fases que debe cumplir todo sistema de software y que aplicaremos:

- PLANIFICACIÓN DE SISTEMAS DE INFORMACIÓN.
- DESARROLLO DE SISTEMAS DE INFORMACIÓN.
- MANTENIMIENTO DE SISTEMAS DE INFORMACIÓN.

En cuanto al Proceso de Desarrollo de Sistemas de Información, para facilitar la comprensión y dada su amplitud y complejidad se ha subdividido en cinco subprocesos:

1. Estudio de viabilidad del sistema.
2. Análisis del sistema de información.
3. Diseño del sistema de información.
4. Construcción del sistema de información.
5. Implantación y aceptación del sistema.

Toda esta documentación podemos encontrarla en la página oficial de METRICA V3: http://www.csi.map.es/csi/metrica3/index.html.

4.1. Herramientas de modelado usadas

Para realizar los diagramas de secuencia del sistema, diagrama de casos de uso, modelo conceptual de datos y diagrama de clases, se han utilizado varias herramientas, las cuales son:

4.1.1. UMLet

UMLet es una herramienta para dibujar diagramas y una herramienta de código abierto de Java basado en UML. Se puede utilizar como aplicación Java independiente, o como un plugin integrado en Eclipse, y se puede descargar del siguiente enlace: http://www.umlet.com/changes.htm. Su interfaz es sencilla de manejar, simplemente tiene un panel de objetos que se pueden utilizar, cambiándoles las propiedades y colores para poner los diagramas a nuestro gusto. También podemos exportar nuestros diagramas en varios formatos, entre ellos JPG, PNG, PDF...

4.1.2. Dia

Dia es otra herramienta para dibujar diagramas pero más completa que UMLet, ya que se puede utilizar para dibujar Bases de Datos, entre otros. Se puede descargar desde aquí: http://projects.gnome.org/dia/.

4.2. Requisitos del sistema y estudio de viabilidad.

Antes de comenzar el proyecto hay que realizar un estudio de viabilidad y analizar los requisitos necesarios para la realización del proyecto. Para ello, debemos realizar una recogida de información y establecer los requisitos de nuestro sistema, los cuales son los siguientes:

- Búsqueda de artículos.
- Creación de carrito virtual.
- Realización de transacción y compra.

4.3. Análisis

En la parte de análisis conseguiremos la especificación detallada del sistema, a través de un análisis de requisitos funcionales que necesite para satisfacer unas necesidades concretas.

Para el análisis y desarrollo del proyecto es muy importante realizar el modelado tanto de análisis como de diseño.
Para este proyecto se ha utilizado el modelo de ciclo de vida de Software, modelo en Espiral[5] (Desarrollar aplicaciones a partir de otro software ya existente), puesto que mi proyecto se centra en la reingeniería de una composición en lenguaje WS-BPEL 2.0.

Modelo en espiral

En el modelo en espiral, el software se desarrolla en una serie de versiones incrementales. En las primeras iteraciones, la versión del software puede ser un prototipo y en las siguientes iteraciones se obtienen versiones cada vez más completas del software. El modelo, representado mediante una espiral, define una serie de actividades o regiones de tareas. Generalmente, existen entre tres y seis regiones de tareas.

Comenzaremos el análisis del sistema realizando el análisis de casos de uso. Un caso de uso especifica un conjunto de secuencias de acciones, incluyendo variantes, que el sistema puede ejecutar y que produce un resultado observable de valor para un particular actor. Un caso de uso especifica el comportamiento deseado del sistema. Representan los requisitos funcionales del sistema. Un actor es representan roles que pueden jugar los usuarios de los casos de uso al interactuar con el sistema. Un caso de uso describe un conjunto de secuencias de interacciones entre actores y el sistema (escenarios). Un escenario es una instancia de un caso de uso, es una historia particular de un caso de uso.

Para representar los casos de uso de nuestro sistema, nos apoyaremos en diagramas y esquemas. Identificaremos en él Actores, casos de uso y descripción de cada caso de uso.

4.3.1. Casos de uso.

Diagrama de casos de uso:

En la figura 4.1 se muestran los casos de usos asociados a los requisitos del sistema.

Descripción de casos de uso.

Caso de uso: Iniciar sesión.

- **Actor:** Usuario (en este caso es un usuario y a la vez cliente).
- **Precondición:** el usuario debe estar registrado previamente en el sistema.
- **Postcondición:** el usuario inicia sesión en el sistema.
- **Escenario principal:**
  1. El usuario desea buscar un libro e introduce el id de cliente y la contraseña.
  2. El sistema comprueba que la contraseña es correcta y que existe el cliente.
- **Escenario alternativo:**

  2a. La contraseña introducida es incorrecta o el usuario no es correcto:

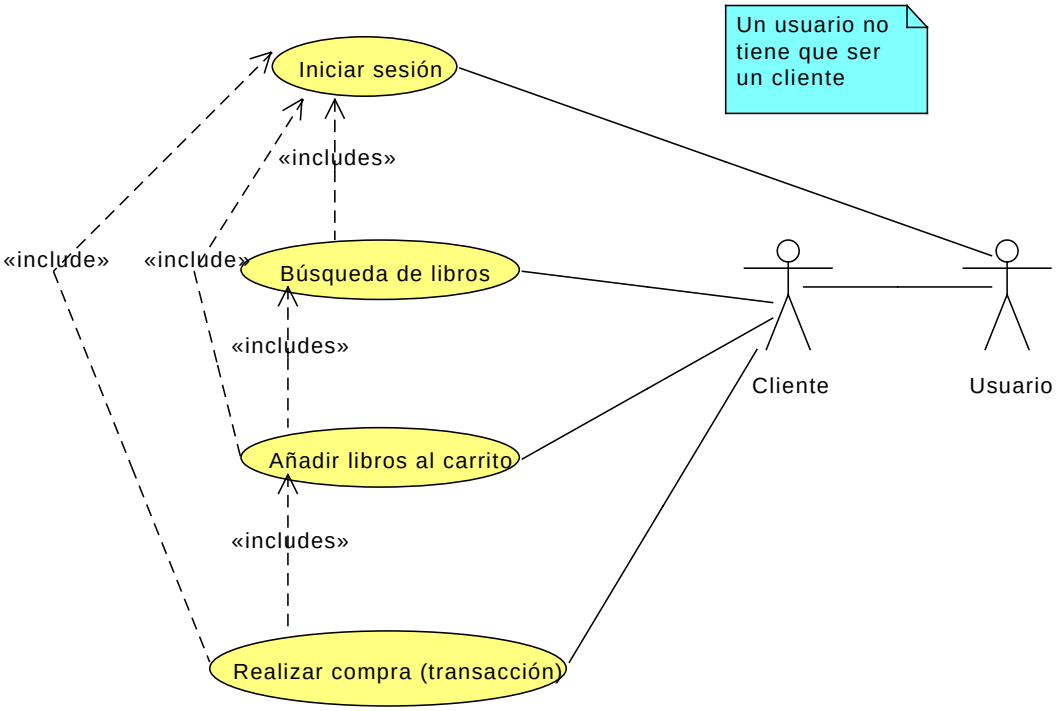


Figura 4.1: Diagrama de casos de uso del Sistema

1. El sistema indica el error y cancela el inicio de sesión.

*a. El usuario cancela el inicio de sesión.

Caso de uso: Búsqueda de libros

- **Actor:** En este caso es el Cliente, ya que se necesita estar identificado para acceder a la búsqueda.
- **Descripción:** El usuario desea realizar una búsqueda de libros en la tienda virtual.
- **Precondiciones:** El usuario debe haber iniciado sesión («include» caso de uso Iniciar sesión).
- **Postcondiciones:** El sistema muestra el resultado de la búsqueda.
- **Escenario principal:**
  1. El usuario introduce los datos de la búsqueda.
  2. El sistema busca el libro con los datos proporcionados.
  3. El sistema muestra una lista de libros encontrados (resultados búsqueda).
- **Escenarios alternativos:**

  *a. El cliente sale de la sesión.

  2a. El sistema da error de búsqueda
      1. El sistema muestra un código de error y cancela la búsqueda.

  2b. El sistema no encuentra el libro reclamado.
      1. El sistema muestra un código de error y cancela la búsqueda.

Caso de uso: Añadir libros al carrito.

- **Actor:** En este caso es el Cliente, ya que se necesita estar identificado para introducir elementos en el carrito virtual.
- **Descripción:**El usuario introduce libros en el carrito de compra virtual.
- **Precondiciones:** El usuario debe haber iniciado sesión («include» caso de uso Iniciar sesión). El usuario debe haber hecho previamente una búsqueda de libros («include» caso de uso Búsqueda de libros).
- **Postcondiciones:** se crea un carrito de compra con todos los libros que van a ser comprados por el cliente.
- **Escenario principal:**
  1. El usuario desea almacenar libros en un carrito virtual.
  2. El sistema crea un carrito de compra vacío y le introduce los artículos deseados.
  3. El usuario obtiene el carrito relleno de artículos.

- **Escenarios alternativos:**
  - *a. El cliente cancela el caso de uso.
  - 2a. El sistema da error al rellenar el carrito.
    1. El sistema muestra un código de error y cancela el caso de uso.
  - 2b. El número de libros supera el número máximo de artículos permitidos en un carrito.
    1. El sistema muestra un código de error y cancela el caso de uso.

Caso de uso: Realizar compra (transacción).

- **Actor:** Usuario (cliente).
- **Descripción:**El cliente realiza la compra de sus libros.
- **Precondiciones:** El cliente debe haber iniciado sesión («include» caso de uso Iniciar sesión). El usuario debe haber hecho previamente una búsqueda de libros («include» caso de uso Búsqueda de libros). El usuario debe haber creado previamente un carrito de compra («include» caso de uso Añadir libros al carrito).
- **Postcondiciones:** El cliente realiza su compra con éxito.
- **Escenario principal:**
  1. El cliente envía los datos del carrito y empieza la transacción.
  2. El sistema realiza la operación de compra y envía los datos de la transacción a un sistema externo.
- **Escenarios alternativos:**
  - *a. El cliente cancela el caso de uso.
  - 2a. El sistema da error en la transacción.
    1. El sistema muestra un código de error y cancela el caso de uso.

4.3.2. Modelo conceptual de datos del sistema.

El objetivo de esta frase es la identificación de las necesidades de información de cada uno de los procesos que conforman el sistema de información con el fin de obtener un modelo de datos que contemple todas las entidades, relaciones y atributos necesarios para dar respuesta a dichas necesidades. Se ha usado el modelo de datos de: Diagrama de clases. Este modelado incluye: clases, asociaciones, atributos, restricciones de seguridad... Se trata de uno de los modelos más comunes en el modelado de sistemas orientados a objetos. Un diagrama de clases muestra un conjunto de clases, interfaces y colaboraciones y sus relaciones entre ellos. Las clases representan los bloques de construcción más importantes de cualquier sistema orientado a objetos. Una clase es una descripción de un conjunto de objetos que comparten los mismos atributos, operaciones, relaciones y semántica. Esta notación es independiente de cualquier lenguaje

de programación. No es necesario mostrar todas las características. A veces las clases tienen tantas características, que no es conveniente mostrarlas todas. En la figura 4.2 de la página 50 se muestra el modelo conceptual de datos.

4.3.3. Modelo de comportamiento del sistema: diagramas de secuencia y contratos de las operaciones.

Los diagramas de secuencia del sistema muestran la secuencia de eventos entre los actores y el sistema y permiten identificar las operaciones del sistema. El contrato de las operaciones describen el efecto de las operaciones del sistema.

Diagramas de secuencia del sistema:

Caso de uso: Iniciar sesión

El diagrama de secuencia del sistema para este caso de uso sería el siguiente:

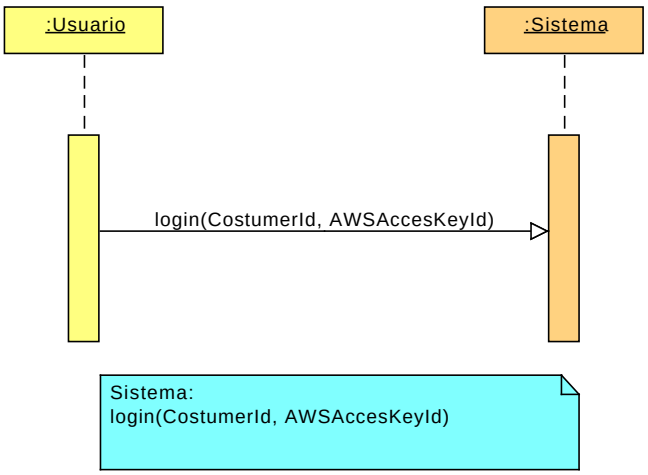


Figura 4.3: Diagrama de secuencia del sistema - Inicio sesión

Caso de uso: Búsqueda de libros

El diagrama de secuencia del sistema para este caso de uso sería el siguiente:

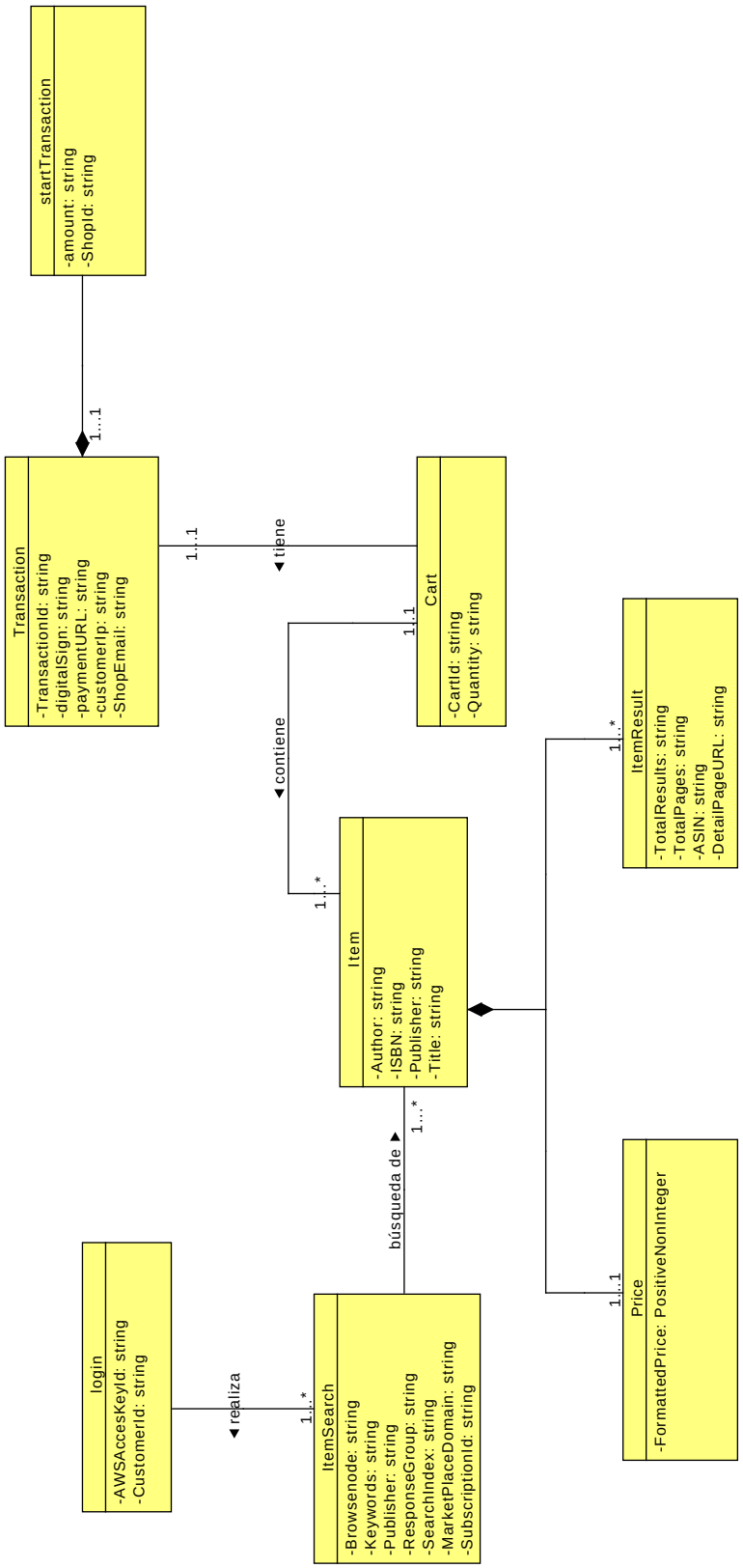


Figura 4.2: Modelo conceptual de datos del sistema

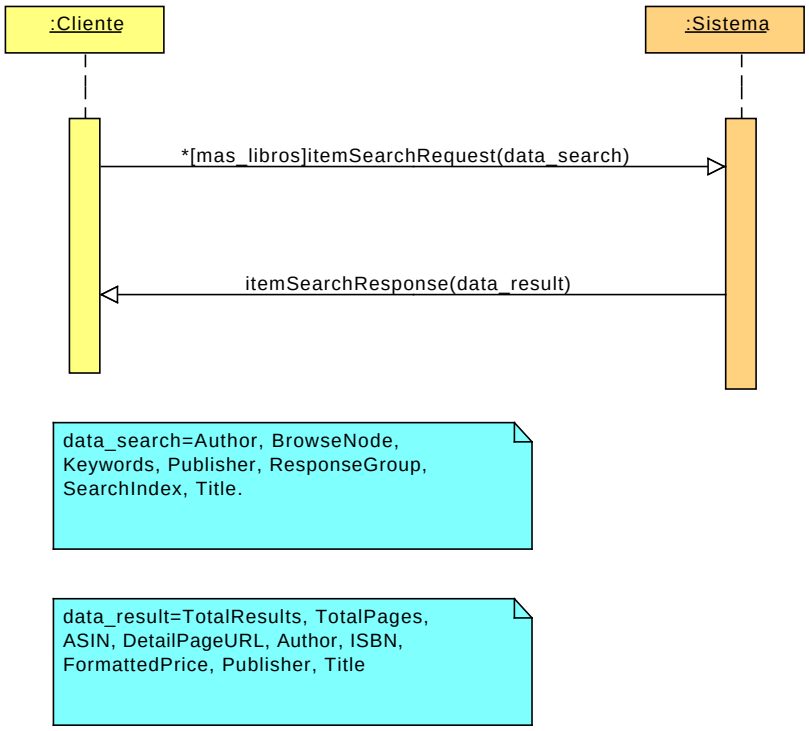


Figura 4.4: Diagrama de secuencia del sistema - Búsqueda de libros

Caso de uso: Añadir libros al carrito

El diagrama de secuencia del sistema para este caso de uso sería el siguiente:

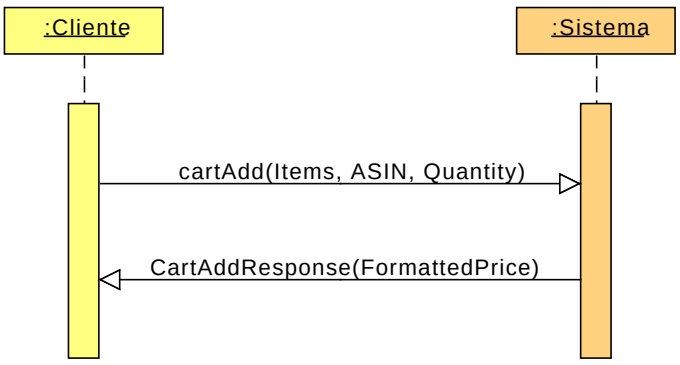


Figura 4.5: Diagrama de secuencia del sistema - Añadir libros al carrito

Caso de uso: Realizar compra (transacción)

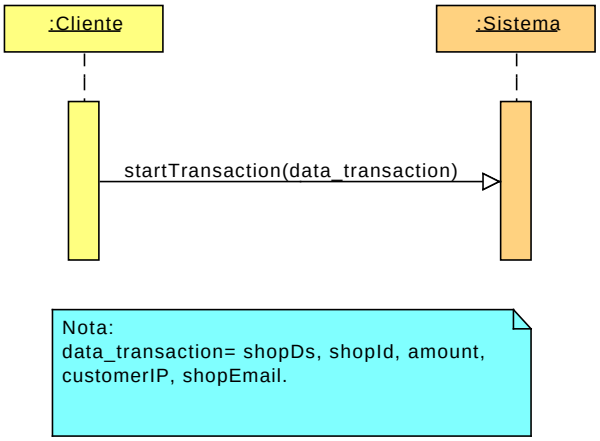


Figura 4.6: Diagrama de secuencia del sistema - Realizar compra (Transacción)

4.3.4. Contrato de las operaciones:

A través del contrato de las operaciones, se describe el comportamiento del sistema en función de los cambios de estado de los objetos del Modelo Conceptual de Datos después de la ejecución de una operación del sistema. Los diagramas de secuencia muestran la secuencia de eventos que producen los actores del sistema. Nos ayudan a la identificación de las operaciones del sistema. Los contratos de las operaciones nos permiten describir el efecto de las operaciones del sistema.

Operación: login(CostumerId, AWSAccesKeyId)
Responsabilidades: Iniciar la sesión en el sistema.
Referencias cruzadas: Caso de uso: Iniciar sesión.
Precondiciones: El sistema comprueba que existe un usuario con clave AWSAcceskeyId = w_AWSAccesKeyId y con nombre de usuario CostumerId = w_CostumerId.
Postcondiciones: Inicia sesión en el sistema.

Operación: itemSearchRequest(Author, BrowseNode, Keywords, Publisher, ResponseGroup, SearchIndex, Title)
Responsabilidades: busca los libros solicitados por el cliente.
Referencias cruzadas: casos de uso: Búsqueda de libros.
Precondiciones: el sistema busca el libro con las palabras claves dadas por el cliente.
Postcondiciones: el sistema muestra el resultado de la búsqueda.

Operación: cartAdd(ASIN, Quantity, FormattedPrice)
Responsabilidades: añadir los libros en un carrito virtual.
Referencias cruzadas: caso de uso: Añadir libros al carrito.
Precondiciones: el sistema recibe los datos y comprueba que no existe aun un carrito creado para ese cliente.
Postcondiciones: crea una instancia del carrito virtual llamado *Cart* y le devuelve al cliente el carrito virtual creado.

Operación: startTransaction(shopDs, shopId, amount, customerIP, shopEmail)
Responsabilidades: Realiza la compra del cliente.
Referencias cruzadas: caso de uso: Realizar compra (transacción).
Precondiciones: el sistema envía los datos del pago a un Servicio Web exterior.
Postcondiciones: se realiza la transferencia y se efectúa la compra.

4.4. Diseño

En el proceso de análisis hemos obtenido los requisitos del sistema y la especificación de qué ha de hacer el sistema, así como las operaciones que realizará el sistema. En esta parte nos centramos en el diseño del software. El objetivo del proceso de Diseño del Sistema es la definición de la arquitectura del sistema y del entorno tecnológico que le va a dar soporte, junto con la especificación detallada de los componentes del sistema de información. Las actividades de este proceso se agrupan en dos grandes bloques. El primer bloque consiste en la Definición de la Arquitectura del Sistema se realiza la descripción de los subsistemas y componentes de un sistema software y las relaciones entre ellos. El segundo bloque es el diseño del sistema software que comprende el diseño de los componentes del Sistema. Es un proceso en el cual se traducen los requisitos en un modelo o representación del sistema que se va a construir. Inicialmente, el diseño se representa a un alto nivel de abstracción y a medida que se realizan iteraciones, el refinamiento siguiente lleva a representaciones del diseño de mucho menor nivel de abstracción.

4.4.1. Arquitectura del sistema software.

La arquitectura donde se encuentra ASTRO sería GAmera, ya que formaría parte, de algún modo de ella, ya que la composición de la tienda virtual sirve como entrada a GAmera y todo lo que le rodea. En la figura 4.7 podemos ver la arquitectura de GAmera con el nuevo generador de casos de prueba.
En el apéndice B se habla más a fondo sobre la arquitectura de esta herramienta y los elementos que la componen.

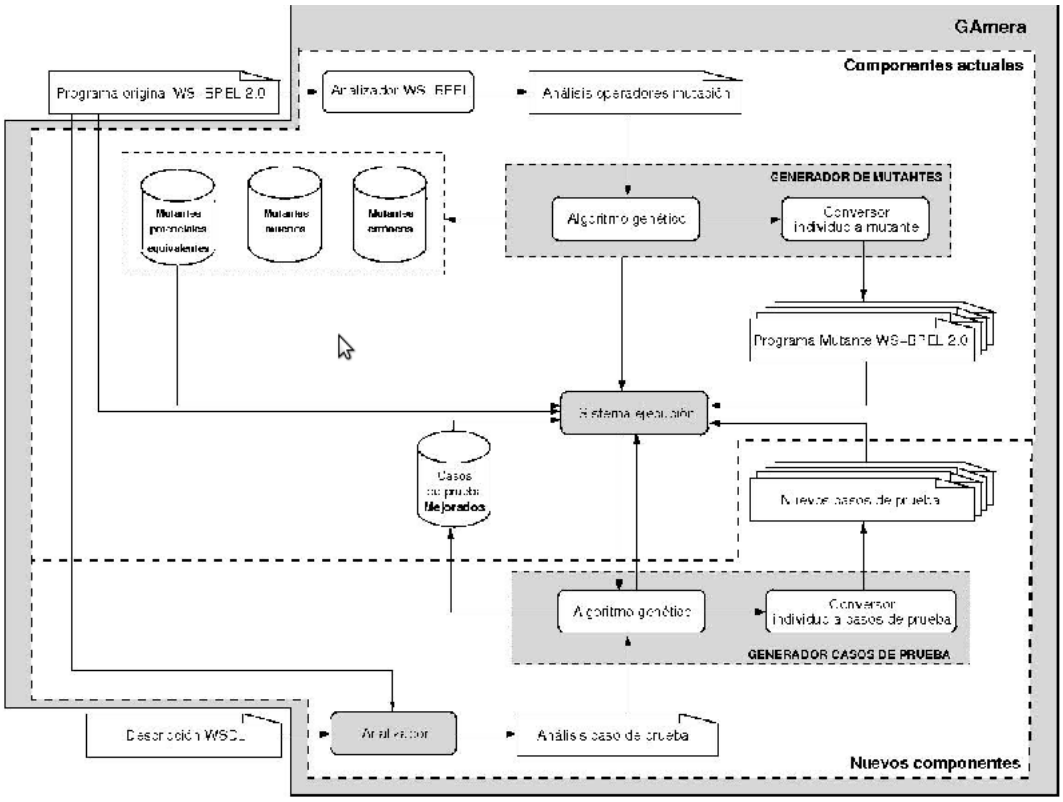


Figura 4.7: Arquitectura de GAmera

4.4.2. Diagrama de Interacción (Diagrama de secuencia)

En el diagrama de interacción se describe cómo los objetos colaboran entre sí para realizar cierta actividad (interacción). Existen dos tipos de Diagramas de Interacción:

- Diagramas de Secuencia: destacan la ordenación temporal de los mensajes.
- Diagramas de Colaboración: destacan la organización estructural de los objetos participantes.

4.4.3. Diagrama de clases de diseño

Mediante los diagramas de secuencia definiremos la interacción entre las clases de objetos en respuesta a los eventos producidos en el sistema. Utilizaremos los diagramas de secuencia ya que éstos dan a conocer de una forma efectiva el orden de los mensajes entre objetos y eventos. Utilizaremos cajas para representar objetos, clases y multiobjetos. La figura 4.8 de la página 64 nos muestra el diagrama de clases de diseño para ASTRO.

4.4.4. Diagramas de secuencia de diseño

Diagrama de secuencia de diseño. Caso de uso: Iniciar sesión

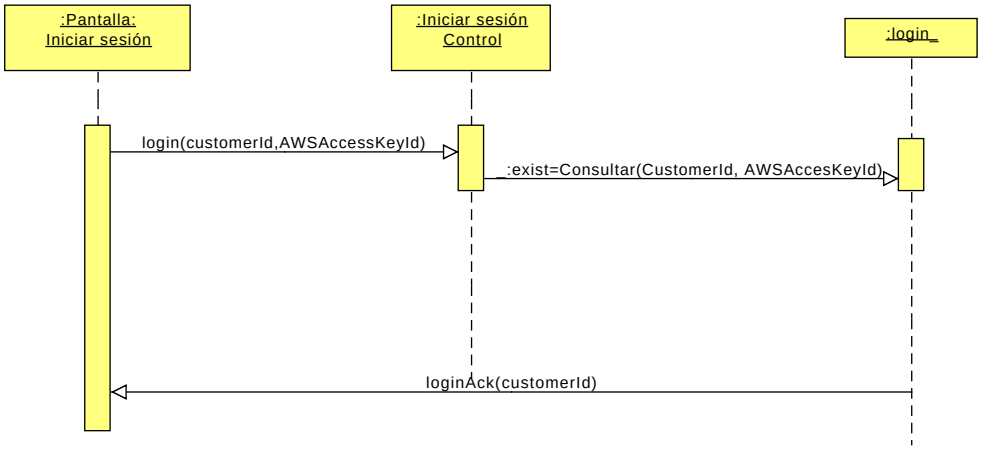


Figura 4.9: Diagrama de secuencia de Diseño -Iniciar sesión

Diagrama de secuencia de diseño. Caso de uso: Búsqueda de libros

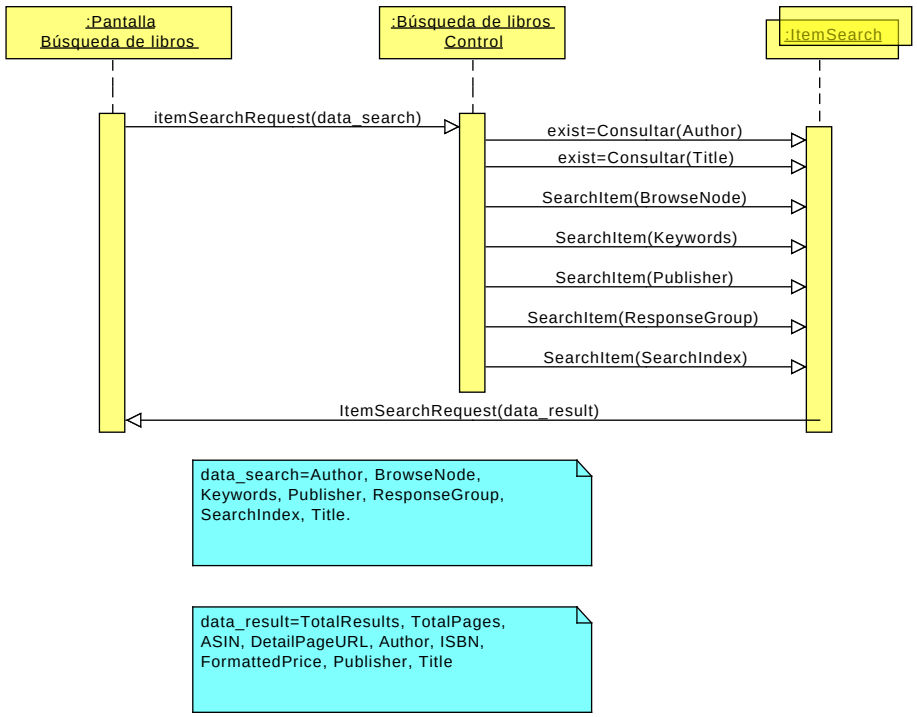


Figura 4.10: Diagrama de secuencia de Diseño -Búsqueda de libros

Diagrama de secuencia de diseño. Caso de uso: Añadir artículos al carrito

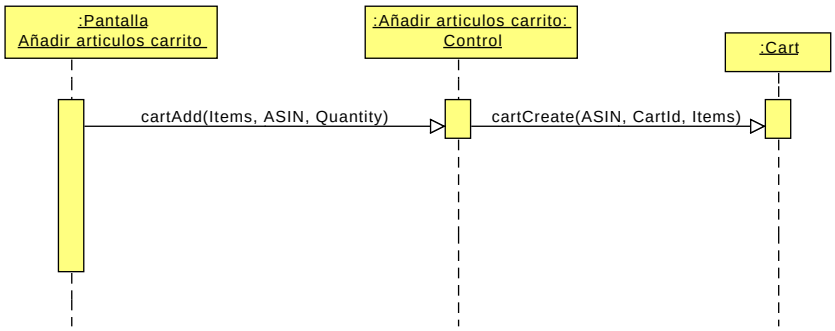


Figura 4.11: Diagrama de secuencia de Diseño -Añadir artículos al carrito

Diagrama de secuencia de diseño. Caso de uso: Realizar compra(transacción)

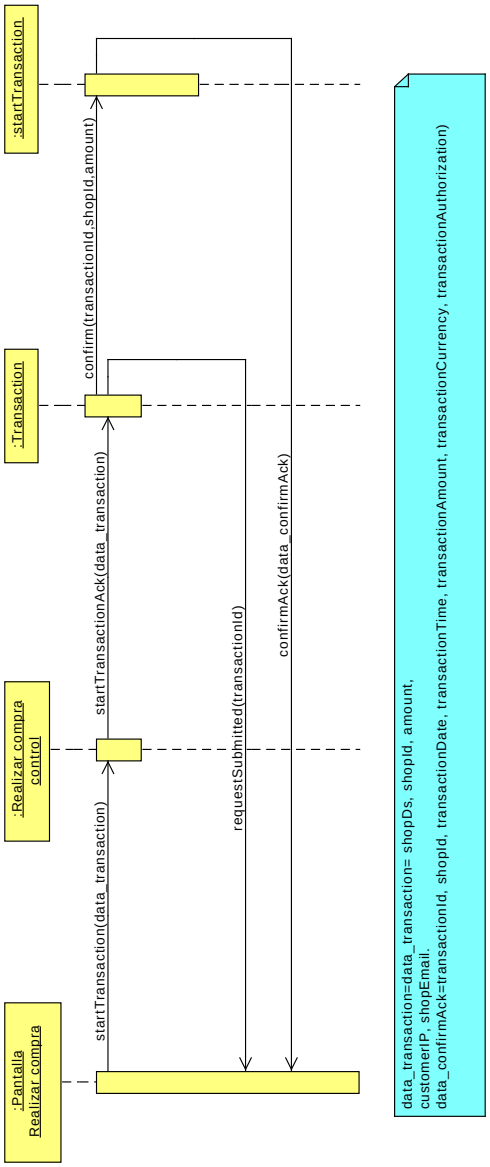


Figura 4.12: Diagrama de secuencia de Diseño -Realizar compra(transacción))

4.5. Pruebas y validación

Son múltiples los resultados que nos ofrece GAmera según las opciones que seleccionemos a lo largo del estudio de mutación que realicemos. Nos vamos a centrar en una de ellas: El tiempo obtenido tras la generación y ejecución de mutantes tanto si escogemos

la opción de generar y ejecutar todos o una selección de los mismos (siempre pondríamos un mismo porcentaje de selección). Haremos un estudio de los resultados que se obtienen con cada una de las composiciones.

4.5.1. Resultados y tiempos

A continuación se realizará un estudio de los tiempos de ejecución de cada una de las composiciones que unen ASTRO, y veremos el porcentaje de mutantes vivos y muertos, así podremos decir si tenemos un buen conjunto de casos de pruebas, en cuanto a calidad de casos de prueba se refiere.
Para realizar este estudio se ha procedido ha realizar los pasos descritos en el apéndice B, capítulo B.5. Finalmente, después de realizar esos pasos, procedemos a realizar un informe estadístico del porcentaje de mutantes muertos, vivos, etc. Para ello utilizamos el guión escrito en *python* llamado `classify-mutants`, implementado dentro del Grupo de Investigación. Este script escrito en *python*, realiza todo el análisis de GAmera de una pasada, y guarda los resultados en un fichero.
Para ejecutar este guión python, primeramente debemos crear un enlace simbólico hacia este guión de la siguiente forma:

```
1 $ ln -l ~/sources-fm/trunk/scripts/classify-mutants
```

O donde tengamos la copia de trabajo.
Después, introduciremos la siguiente orden por la consola:

```
1 ./classify-mutants -r 10 -f ficherosalida.cvs ficheroBPTS ficheroBPEL
```

El segundo parámetro en qué intervalo de tiempo medido en segundos se reiniciará *ActiveBPEL*. Por defecto le dejamos 10. Ahora veamos los resultados que produce cada composición.

Resultados de *ASTROBookSearch*

La tabla 4.2 muestra el porcentaje de mutantes vivos, muertos o erróneos, es decir, podemos ver si nuestros casos de prueba son suficientes, si son buenos y la composición cumple sus objetivos y la calidad de los mismos. Podemos ver en la tabla 4.1, los operadores de mutación que se pueden aplicar a esta composición[1].

Tenemos algunos mutantes nó validos como podemos observar en el porcentaje, es decir, error de despliegue. Estos errores de despliegue no se han podido solventar, debido

[1] Como podemos ver en el apéndice B, sección B.5.2, la representación del individuo o mutante sería de la siguiente manera: (Operador, Localización, Atributo)

Cuadro 4.1: Operadores de mutación que se pueden aplicar en *ASTROBookSearch*

Operador	**Número de instancias en las que se aplica**	**Valores**
ISV	0	1
EAA	1	4
EEU	0	1
ERR	2	5
ELL	0	1
ECC	136	1
ECN	24	4
EMD	0	2
EMF	0	2
ACI	0	1
AFP	0	1
ASF	7	1
AIS	0	1
AIE	2	1
AWR	1	1
AJC	0	1
ASI	11	1
APM	2	1
APA	0	1
XMF	0	1
XMC	0	1
XMT	0	1
XTF	0	1
XER	0	1
XEE	0	1

Cuadro 4.2: Estudio estadístico para la composición *ASTROBookSearch*

Porcentajes	
Mutantes muertos	$51,11\,\%$
Mutantes válidos	$92,78\,\%$
Mutantes repetidos	$0,00\,\%$
Total mutantes	291
Tiempos de ejecución (en segundos)	
Ejecución composición original	$10,97$
Analizador(operadores a aplicar)	$6,18$
Generación mutantes	$735,98$
Comparación(mutantes con composición original)	$4569,13$
Tiempo total	$5322,26$

a que si modificamos algo del código del proceso BPEL, ya estaríamos cambiando la lógica de ASTRO original. Esto también ha ocasionado que siga habiendo mutantes vivos.

Un ejemplo de error de desplegado, puede ser el siguiente. Al aplicar el operador *ASF*, que cambia una actividad `sequence` por una actividad `flow`, sobre la composición, si la instancia en la que se aplica es justo al comienzo del proceso BPEL, entonces dará un error de desplegado, puesto que todo proceso en BPEL debe empezar por una secuencia `sequence`. Como podemos ver en la figura 4.13 de la página 65, se ha cambiado la actividad, y como es el comienzo del proceso BPEL, da error de despliegue.

Resultados de *ASTROBookCart*

Podemos ver en la tabla 4.3, los operadores de mutación que se pueden aplicar a esta composición. La tabla 4.4 muestra el porcentaje de mutantes vivos, muertos o erróneos, es decir, muestra si los casos de prueba son suficientes y tienen calidad.

Cuadro 4.3: Operadores de mutación que se pueden aplicar en *ASTROBookCart*

Operador	**Número de instancias en las que se aplica**	**Valores**
ISV	6	1
EAA	1	4
EEU	0	1
ERR	4	5
ELL	0	1
ECC	101	1
ECN	10	4
EMD	0	2
EMF	0	2
ACI	0	1
AFP	0	1
ASF	10	1
AIS	0	1
AIE	3	1
AWR	1	1
AJC	0	1
ASI	42	1
APM	6	1
APA	0	1
XMF	0	1
XMC	0	1
XMT	0	1
XTF	0	1
XER	0	1
XEE	0	1

Cuadro 4.4: Estudio estadístico para la composición *ASTROBookCart*

Porcentajes	
Mutantes muertos	$67,58\,\%$
Mutantes válidos	$93,99\,\%$
Mutantes repetidos	$0,00\,\%$
Total mutantes	233
Tiempos de ejecución (en segundos)	
Ejecución composición original	$20,42$
Analizador(operadores a aplicar)	$6,48$
Generación mutantes	$572,68$
Comparación(mutantes con composición original)	$4867,18$
Tiempo total	$5466,76$

Resultados de *ASTROBookBank*

Podemos ver en la tabla 4.5, los operadores de mutación que se pueden aplicar a esta composición. La tabla 4.6 muestra el porcentaje de mutantes vivos, muertos o erróneos, es decir, muestra si los casos de prueba son suficientes y tienen calidad.

Cuadro 4.5: Operadores de mutación que se pueden aplicar en *ASTROBookBank*

Operador	**Número de instancias en las que se aplica**	**Valores**
ISV	0	1
EAA	0	4
EEU	0	1
ERR	0	5
ELL	0	1
ECC	0	1
ECN	0	4
EMD	0	2
EMF	0	2
ACI	0	1
AFP	0	1
ASF	7	1
AIS	0	1
AIE	3	1
AWR	0	1
AJC	0	1
ASI	15	1
APM	2	1
APA	0	1
XMF	0	1
XMC	0	1
XMT	0	1
XTF	0	1
XER	0	1
XEE	0	1

Cuadro 4.6: Estudio estadístico para la composición *ASTROBookBank*

Porcentajes	
Mutantes muertos	$92,31\,\%$
Mutantes válidos	$96,30\,\%$
Mutantes repetidos	$0,00\,\%$
Total mutantes	27
Tiempos de ejecución (en segundos)	
Ejecución composición original	$7,26$
Analizador(operadores a aplicar)	$3,45$
Generación mutantes	$42,92$
Comparación(mutantes con composición original)	$456,43$
Tiempo total	$510,08$

Resultados de *ASTROBookStore*

Esta composición es la que engloba a las tres anteriores en una sola, por lo tanto, esta composición es dónde más operadores de mutación pueden aplicarse. Podemos ver en la tabla 4.7 los operadores que pueden aplicarse a esta composición en WS-BPEL. También en la tabla 4.8 se muestran los valores y tiempos de ejecución, así como el porcentaje de mutantes.

Conclusiones

Podemos afirmar que nuestra composición tiene un buen conjunto de casos de prueba, ya que el porcentaje de mutantes válidos y muertos es bastante alto para las tres composiciones, siendo más considerable en la última, *ASTROBookStore*.

Cuadro 4.7: Operadores de mutación que se pueden aplicar en *ASTROBookStore*

Operador	**Número de instancias en las que se aplica**	**Valores**
ISV	1096	1
EAA	2	4
EEU	0	1
ERR	2	5
ELL	0	1
ECC	145	1
ECN	23	4
EMD	0	2
EMF	0	2
ACI	0	1
AFP	0	1
ASF	28	1
AIS	0	1
AIE	0	1
AWR	2	1
AJC	0	1
ASI	744	1
APM	25	1
APA	0	1
XMF	0	1
XMC	0	1
XMT	0	1
XTF	0	1
XER	0	1
XEE	0	1

Cuadro 4.8: Estudio estadístico para la composición *ASTROBookStore*

Porcentajes	
Mutantes muertos	$70,85\,\%$
Mutantes válidos	$99,40\,\%$
Mutantes repetidos	$0,00\,\%$
Total mutantes	2150
Tiempos de ejecución (en segundos)	
Ejecución composición original	$11,65$
Analizador(operadores a aplicar)	$3,2$
Generación mutantes	$3036,83$
Comparación(mutantes con composición original)	$39367,85$
Tiempo total	$42419,53$

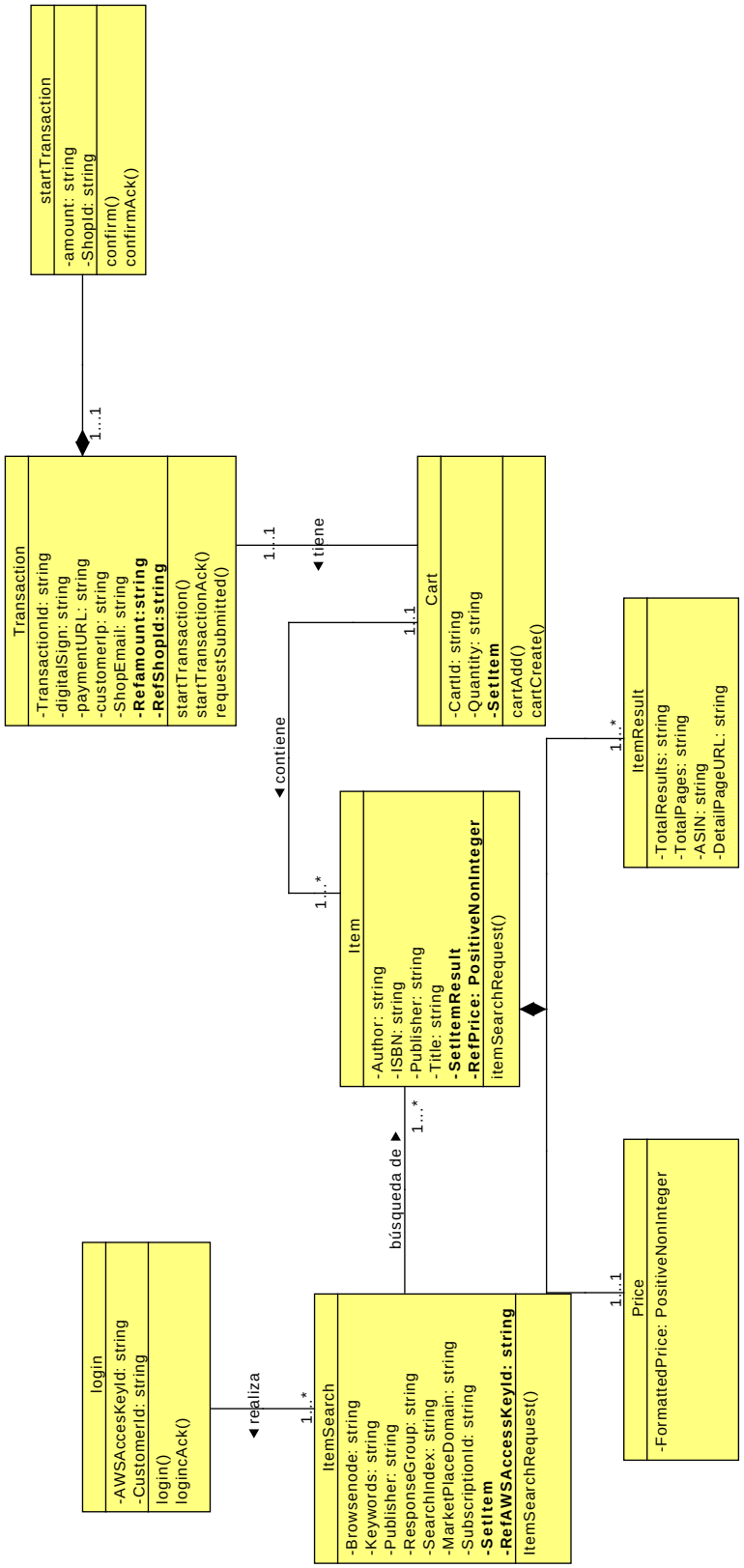


Figura 4.8: Diagrama de clases de diseño

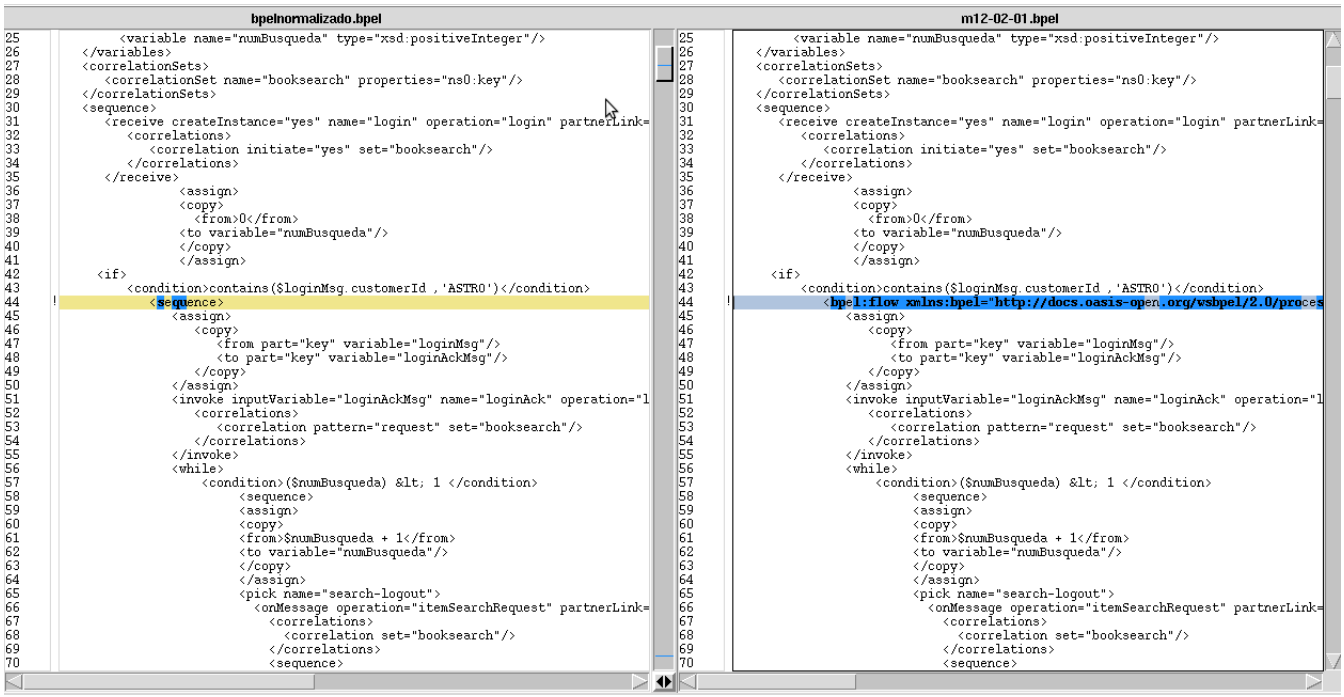


Figura 4.13: Mutante resultado de aplicar el operador ASF a la composición original

5 Resumen

5.1. Resumen.

Este Proyecto Fin de Carrera es llevado a cabo gracias al ofrecimiento por parte de miembros del Grupo de Investigación SPI&FM, Inmaculada Medina Bulo, directora del proyecto, junto a Antonio García Domínguez, guíando en los primeros pasos.
Un problema que se suele encontrar todo el que trabaja con Servicios Web y el lenguaje WS-BPEL es que existen muy pocos ejemplos de mediana complejidad disponibles. Los Servicios Web permiten el intercambio de información en sistemas distribuidos heterogéneos. Forman una arquitectura que permite describir, publicar, descubrir, solicitar e invocar diferentes servicios empleando la infraestructura de internet. El lenguaje WS-BPEL, especifica el comportamiento de un proceso de negocio basado en interacciones con Servicios Web.

El PFC se podría plantear como un proyecto de gestión pero con la característica que está realizado utilizando WS-BPEL 2.0 y Servicios Web, un lenguaje para composiciones de Servicios Web totalmente novedoso. El proyecto tiene también una parte de Investigación, en la que mejoramos la calidad de los casos de prueba realizados.

En la realización del PFC se adquirirán los siguientes conceptos:

- Adquirir los conocimientos del lenguaje WS-BPEL 2.0, y un poco de BPEL4WS 1.1, así como el aprendizaje de los mismos.
- Lenguaje WSDL 1.1, mediante el cual se definirá la interfaz de cada composición WS-BPEL con el exterior.
- XMLSchema 1.1, mediante el cual definiremos los tipos de datos que utilizará nuestra composición.
- XPath 1.0, lenguaje para poder realizar consulta de nodos.
- Parte de investigación en la que se conoce a fondo la herramienta del grupo de investigación: GAmera.
- Estudio de los operadores de mutación para WS-BPEL 2.0, sus características y para qué sirve cada uno de ellos.
- Mutación de una composición WS-BPEL.
- Conocimientos de Servicios Web.

5 Resumen

- Análisis y resultados obtenidos tras la mutación de composición con GAmera.
- Con proyectos de grandes envergaduras como este, es adecuado utilizar un sistema de control de versiones como sería subversion (SVN).
- Lenguaje LaTeX, el cuál es otro lenguaje bastante cómodo y elegante para realizar documentación.

El proyecto está enfocado a la reigeniería de una composición ya existente, llamada ASTRO, la cual está en la versión antigua BPEL4WS (Bpel for Web Services) 1.1, por lo que no sigue el estándar y no sería válido para la herramienta del grupo de Investigación SPI&FM. Mi trabajo en esta parte del proyecto es adaptar la composición ASTRO a la nueva versión WS-BPEL 2.0.
El Grupo tiene una potente herramienta, la cual genera automáticamente mutantes para WS-BPEL 2.0, llamada GAmera. Los mutantes son programas que contienen una única diferencia con respecto al programa original. Se generan aplicando al código fuente un conjunto de reglas definidas previamente, los operadores de mutación. Para esta herramienta no existían ejemplos de complejidad bastante grandes, con lo cual ese es también el objetivo de este proyecto, solventar en algo ese problema.

La parte de investigación, consiste en realizar un conjunto de casos de prueba y mejorar la calidad de los mismos, a través de GAmera. El objetivo de GAmera es detectar los errores más comunes de un programador al programar.

Como modelo de ciclo de vida utilizado para la realización del proyecto, se utilizará el modelo en espiral, ya que mi proyecto se centra en desarrollar una aplicación a partir de otro software ya existente.

6 Conclusiones

Tras realizar este Proyecto Fin de Carrera, se han obtenido bastantes conocimientos nuevos, nuevos lenguajes de programación y otra forma de enfocar proyectos de ingeniería, a través de la investigación. También se ha aprendido a manejar nuevas herramientas y tecnologías, como podría ser el control de versiones.

- Hemos obtenido conocimientos sobre las mutaciones, los algoritmos genéticos, el funcionamiento del algoritmo de mutación, el comportamiento y funcionamiento de los operadores de mutación.
- Se han aprendido los lenguajes de programación a fondo implicados en el proyecto, como son WS-BPEL 2.0, BPEL4WS 1.1, WSDL, XPaht, SOAP y XML Schema.
- Se ha aprendido a utilizar el lenguaje LaTeX, ya que es bastante cómodo a la hora de desarrollar gran documentación para un proyecto serio.
- Se ha aprendido a manejar nuevas tecnologías, como son el control de versiones con *Subversion* (conocido como SVN), ya que permite facilidad para llevar en orden las versiones y fases del proyecto a medida que avanza.
- Al finalizar el proyecto, hemos aprendido conocimientos teóricos y prácticos, y hemos puesto en práctica el método en espiral, el estudio de aplicaciones y los conocimientos adquiridos con los lenguajes de programación, con la mutación, algoritmos genéticos, etc.
- Obtener conceptos sobre Servicios Web y todo lo que ello conlleva.
- Demostración experimental y práctica durante la reingeniería de la composición ASTRO, mediante una de las herramientas desarrolladas por el Grupo, GAmera, la cual permite depurar un proceso WS-BPEL detectando errores de programación que no son apreciables con facilidad.
- Mejor conocimiento de entornos como *Netbeans*, o *Eclipse*, así como manejo con el editor integrado para lenguaje BPEL.

Se ha apreciado durante la elaboración y desarrollo de este PFC es necesaria, ya que ayudaría bastante al programador para detectar los errores más comunes al programar, y realizar este trabajo un poco más fácil.

También se ha madurado el auto-aprendizaje de los lenguajes implicados y el comportamiento de GAmera, por supuesto, no sin la ayuda crítica de miembros del Grupo de Investigación, así como de mi tutora del Proyecto en momentos claves. Este PFC

ha sido una reingeniería de cuatro composiciones en BPEL, las cuales componen ASTRO. Pero la parte más dura sin duda ha sido, tras el desarrollo de la adaptación de BPEL4WS 1.1 a WS-BPEL 2.0, la realización de los casos de prueba correspondientes a cada composición, ya que he tenido muchísimos problemas, por ser mensajes asíncronos. Los mensajes asíncronos en los Servicios Web, son un poco más laboriosos por esa incertidumbre de no saber cuándo se recibe el mensaje; el proceso BPEL envía un mensaje asíncrono, y el cliente (en este caso el *mockup*) lo recibe, pero debido a que es asíncrono es un poco más incierto a la hora de elaborar el caso de prueba.

Finalmente, concluir diciendo que se han alcanzado todos los objetivos del proyecto, tanto la adquisición de los conocimientos teóricos, como técnicos, como el aprendizaje de las técnicas de estilo a la hora de elaborar una aplicación la cual ya existía, y el cumplir y mejorar los requisitos del Cliente (en este caso, los componentes del Grupo de Investigación).

7 Manual de Instalación

7.1. Instalación de NetBeans + GlassFish.

7.1.1. Instalación y configuración bajo Windows

Por las facilidades que ofrece para realizar las composiciones Web, una buena elección es el IDE NetBeans que en su versión 'Enterprise' posee un editor de WS-BPEL. Se puede descargar desde http://bits.Netbeans.org/download/6.7/m3/. Es importante y necesario descargarse la versión "ALL" para disponer del editor BPEL. Para mayor facilidad, es conveniente descargarlo en español. Al instalar, hay que tener cuidado con la ruta de almacenamiento, es aconsejable no introducir rutas con espacios en blanco. Para instalar NetBeans, es necesario tener instalado previamente un JDK, que se puede descargar desde http://java.sun.com/javase/downloads/index.jsphttp://java.sun.com/javase/downloads/index.jsp. Primeramente instalamos JDK. Posteriormente, procedemos instalar NetBeans, cuya interfaz puede verse en la figura 7.1.
Es una interfaz no muy difícil de manejar cuando se familiariza con ella. Para adaptar la herramienta y poder utilizar el editor BPEL, accedemos a la barra de herramientas(*Tools*) /Complementos(*Plugins*). Podemos ver este paso en la figura 7.2.

Después de esto, seleccionaremos la pestaña *Available Plugins*, hacemos una búsqueda en "Buscar" e introducimos la palabra "BPEL", o también puede aparecer poniendo la palabra "SOA". Aparecerá el plugin de *SOA* y le damos a instalar. Podemos verlo gráficamente en la figura 7.3.

Una vez instalado satisfactoriamente, reiniciamos el IDE y ya tendremos instalados el plugin, y nos aparecerá como en la figura 7.4.

Ya tendremos disponible en *NetBeans*, la posibilidad de realizar un proyecto */SOA/Bpel Module* aunque no podremos ejecutarlo aún [1].

Instalación de GlassFish

Para poder empezar a "ejecutar" nuestras composiciones en el entorno de NetBeans necesitamos un servidor de aplicaciones que permita modelar el intercambio de mensajes

[1]Para ello utilizaremos la herramienta que nos proporciona el Grupo de Investigación *mutationtool*

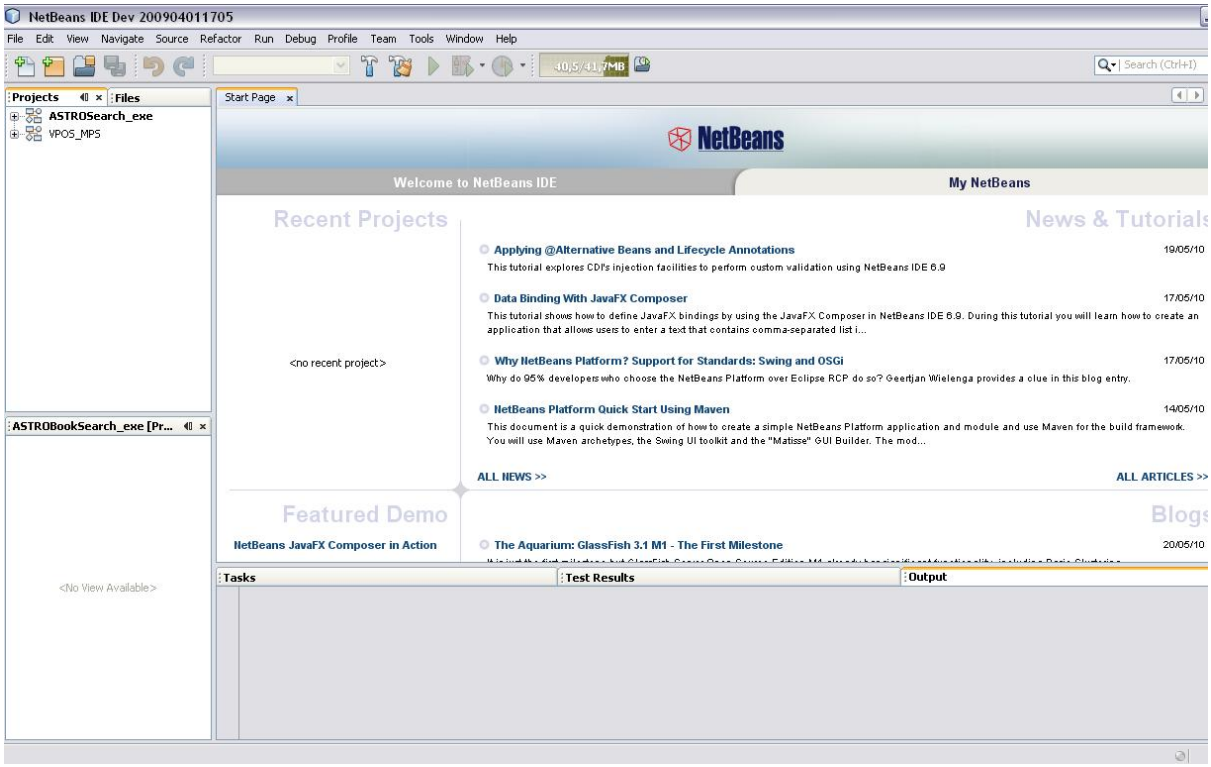


Figura 7.1: Interfaz de NetBeans

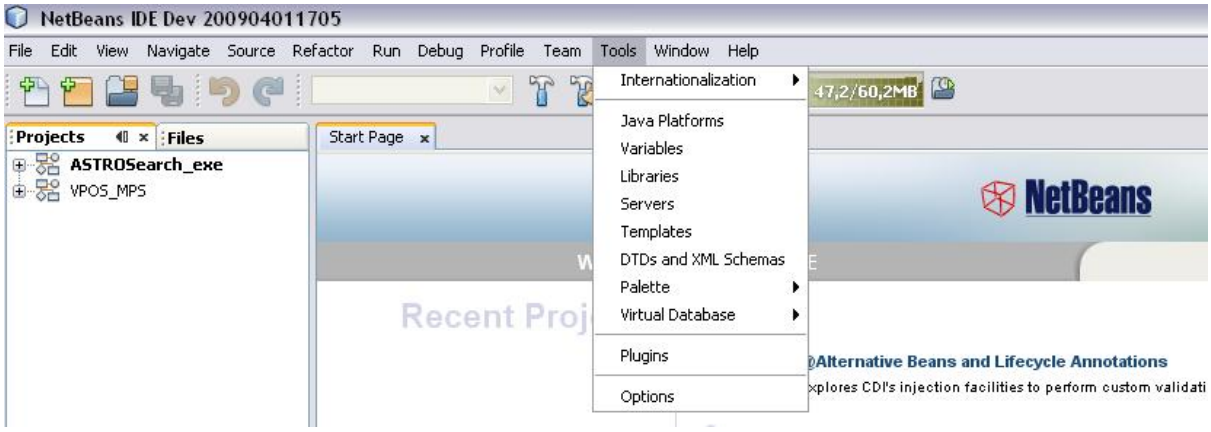


Figura 7.2: Pasos para editor BPEL

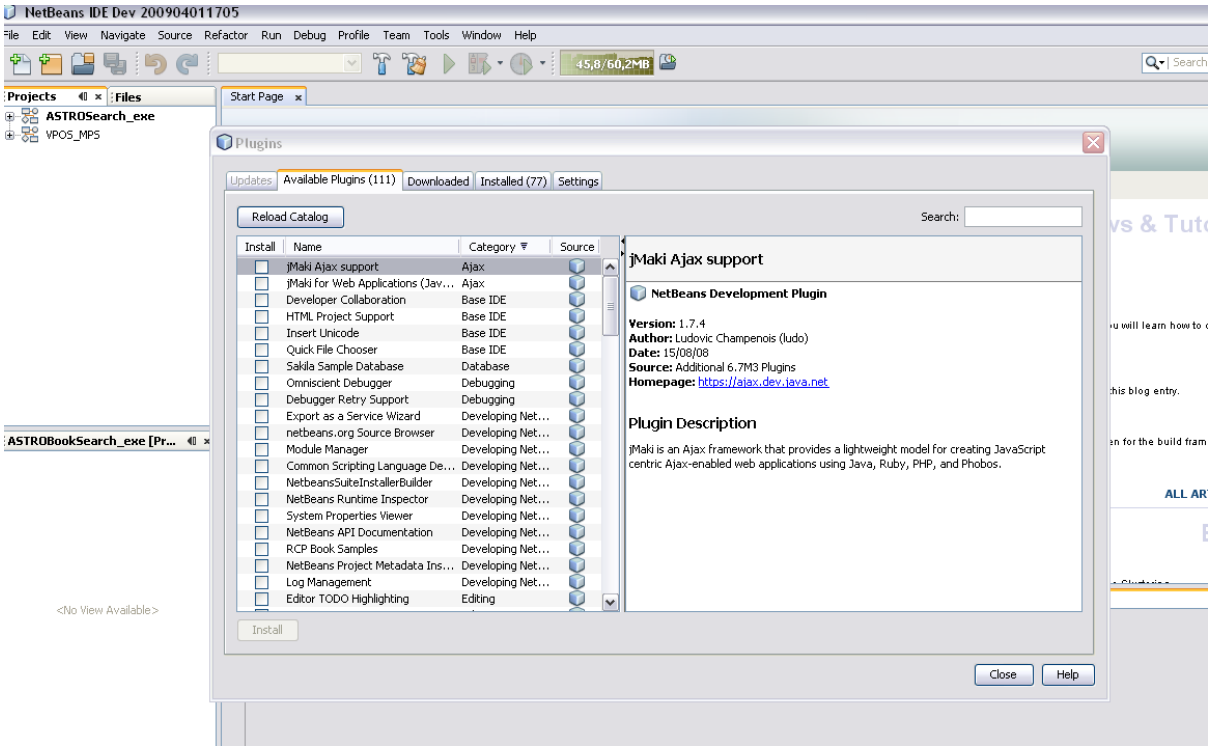


Figura 7.3: Búsqueda del plugin SOA

de un proceso de negocio BPEL. Básicamente, *GlassFish*[16] es un servidor de aplicaciones Java Enterprise Edition 5, es código abierto y sirve para desarrollar aplicaciones empresariales (transaccionalidad, aplicaciones distribuidas, arquitecturas MOM, desarrollo portales WEB, desarrollo SOA, etc).

GlassFish tiene como base al servidor Sun Java System Application Server de Sun Microsystems (en la actualidad comprado por Oracle), un derivado de Apache Tomcat, y que usa un componente adicional llamado Grizzly que usa Java NIO para escalabilidad y velocidad. Esta versión nos será útil para simular nuestros mensajes recibidos y enviados de los WS (Servicios Web), y podemos descargarlo fácilmente desde `https://open-esb.dev.java.net/Downloads.html`. En la figura *fig:gf1* vemos la primera ventana que nos saldrá al instalar GlassFish. El siguiente paso será el directorio donde será instalado GlassFish. Deberemos instalarlo en la carpeta donde previamente se ha instalado NetBeans. Tiene que ser obligatoriamente en este directorio, ya que si no, no podrá hacerse el desplegado de las composiciones. Lo podemos ver en la figura 7.6. Tras descargarlo, lo instalamos (en este proceso, debemos buscar la carpeta donde esta instalado NetBeans y la antigua versión de GlassFish), como se ha dicho anteriormente. Ahora en la figura 7.7 se puede observar la ruta que debe tomar.

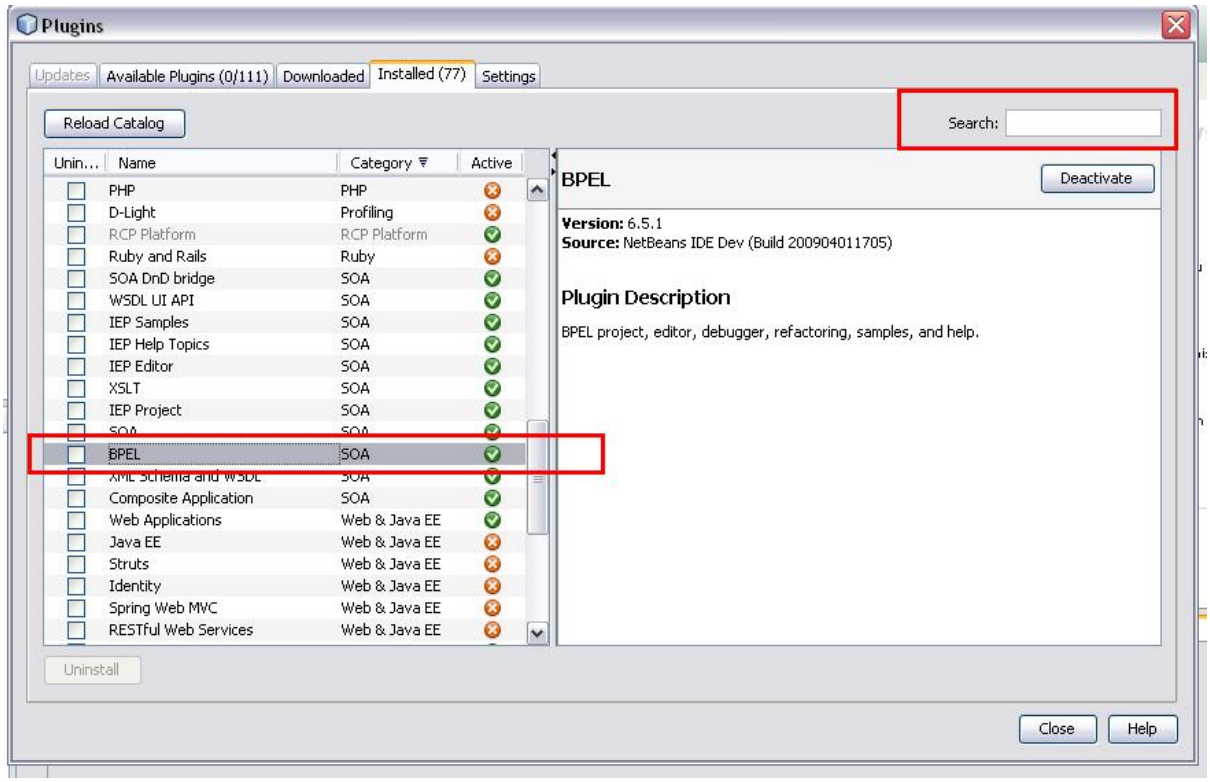


Figura 7.4: Plugin editor BPEL instalado

Figura 7.5: Primer paso en la instalación de GlassFish

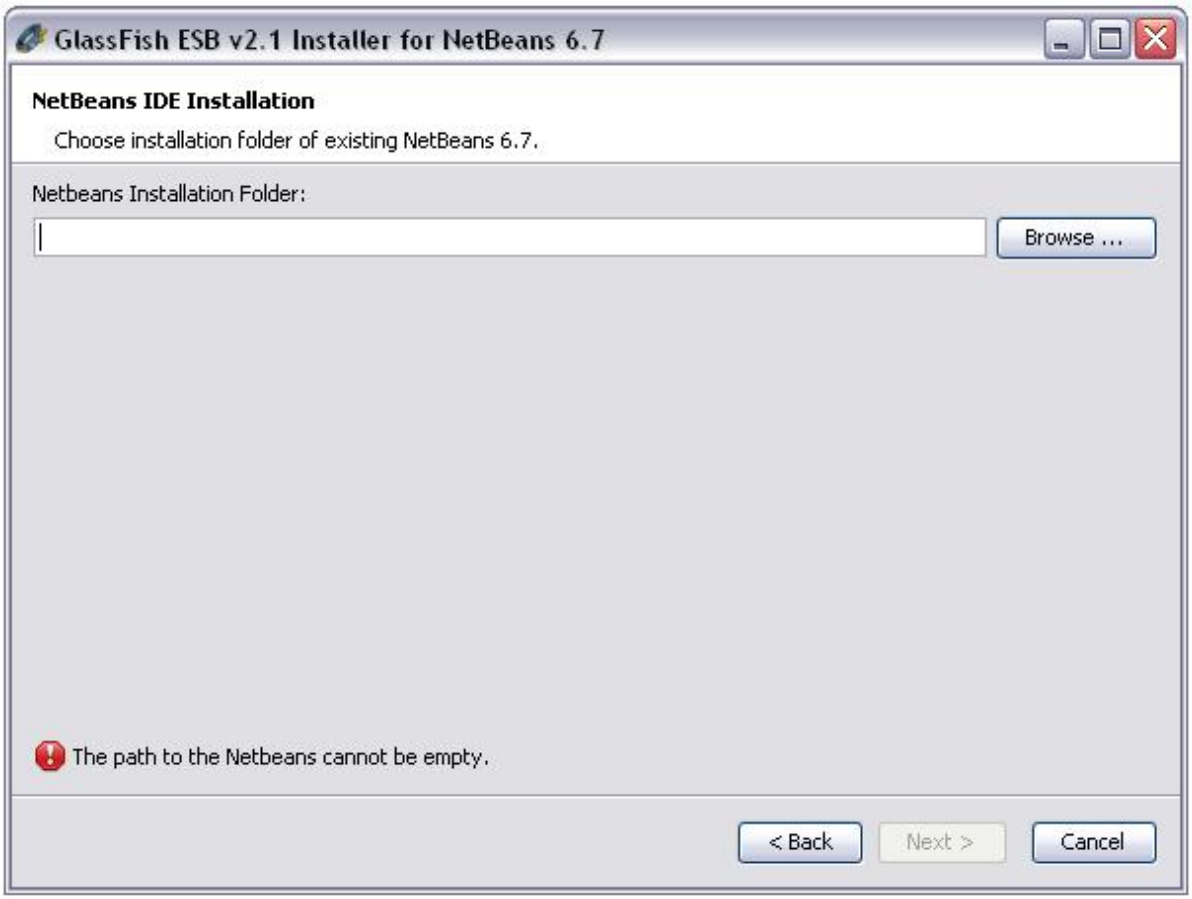


Figura 7.6: Segundo paso en la instalación de GlassFish

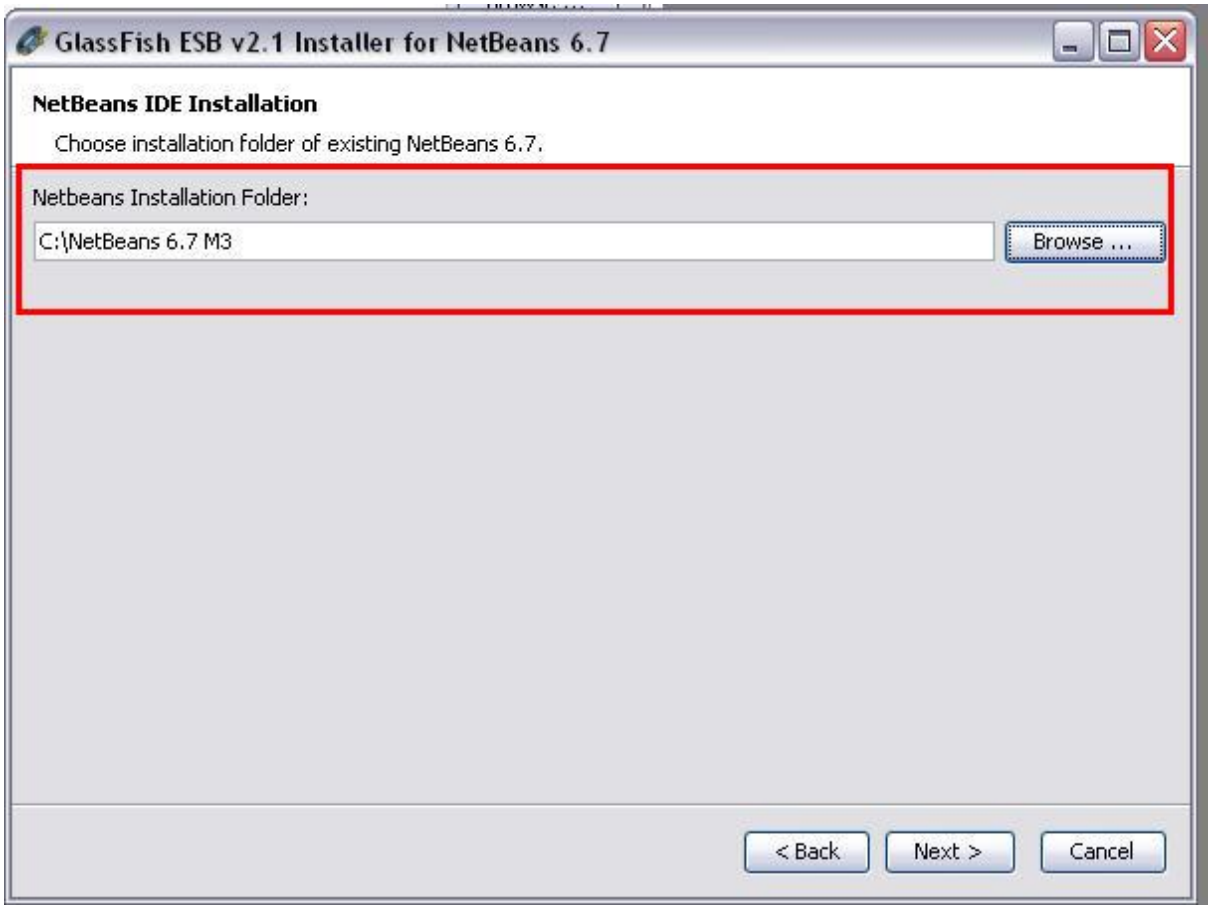


Figura 7.7: Instalación de GlassFish v3

7.2. Instalar GAmera: mutationtool

7.2.1. Instalación bajo Ubuntu 9.10

Para instalar mutationtool, nos tenemos que hacer una copia de trabajo de los repositorios de Redmine:

```
svn co http://neptuno.uca.es/svn/sources-fm/
```

Una vez que nos hemos hecho la copia de trabajo, tendremos las siguientes carpetas: branches, tags, trunk. En realidad solo nos hará falta 'trunk', así que bastará con hacerse una copia de trabajo de esa carpeta:

```
svn co http://neptuno.uca.es/svn/sources-fm/trunk
```

Ahora entramos en la carpeta "scripts", y ejecutamos en la consola el siguiente comando:

```
./install.sh gamera
```

Instalaremos GAmera, que es lo que necesitaremos para ejecutar nuestras composiciones.

7.2.2. Dependencias necesarias

Ahora describiremos las dependencias necesarias para poder ejecutar procesos en BPEL.

Utilizaremos Apache Tomcat. Añadiremos las variables de entorno necesarias. Hay que asegurarse de que las variables de entorno JAVA_HOME y JDK_HOME estén debidamente configuradas:

```
echo \$JAVA\_HOME

echo \$JDK\_HOME
```

Si en alguno de los casos no aparece nada, deberemos añadir la línea correspondiente de estas dos (el valor variará si se usa una distribución distinta de Ubuntu):

```
export JAVA\_HOME=/usr/lib/jvm/java-6-sun

export JDK\_HOME=/usr/lib/jvm/java-6-sun

export CATALINA\_HOME= ~/bin/tomcat5
```

8 Manual de Usuario

8.1. Desarrollo de composiciones con NetBeans

8.1.1. Nuevo proyecto en el IDE

Para realizar un nuevo proyecto BPEL, procedemos de la siguiente manera: *File/New Project...* Elegimos el tipo *SOA*, dentro de SOA marcamos BPEL Module:

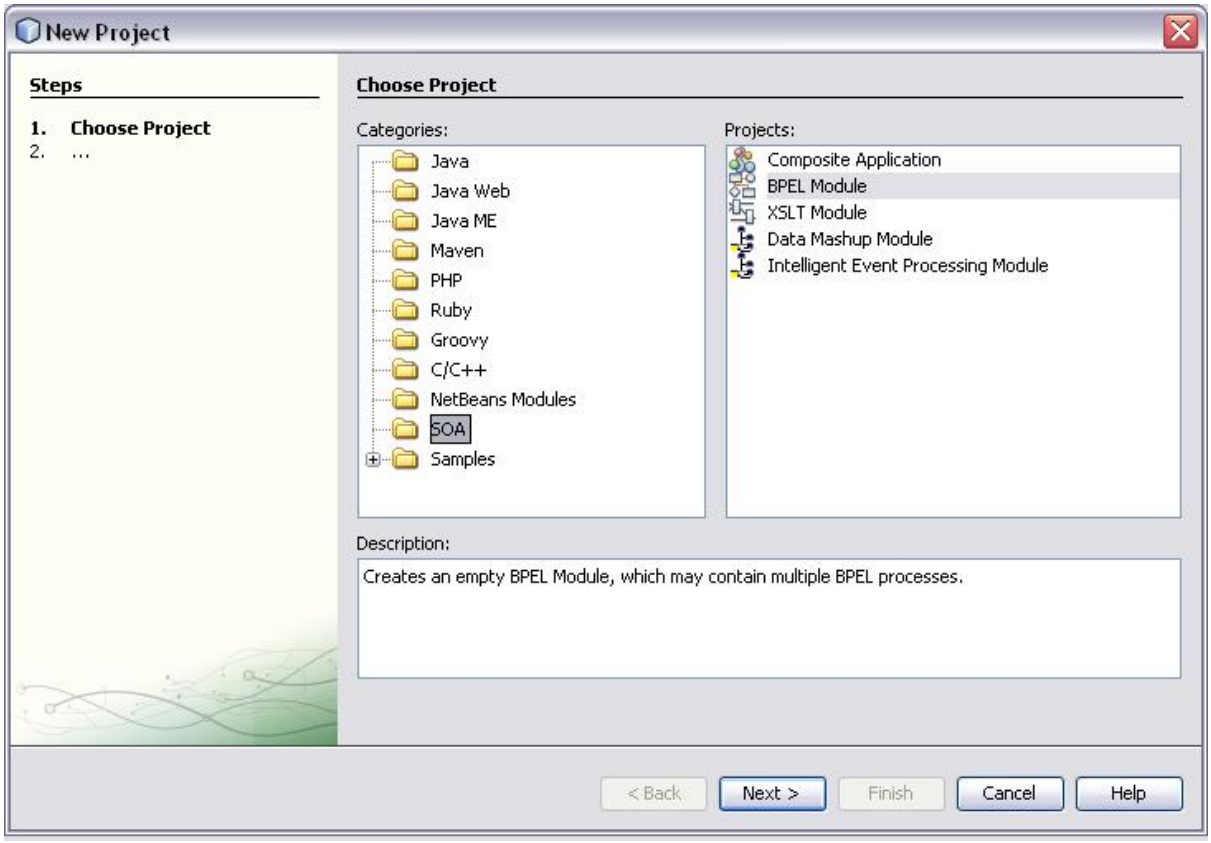


Figura 8.1: Nuevo proyecto BPEL con NetBeans

Para añadir un nuevo fichero .bpel al proyecto para realizar nuestras composiciones, o un archivo .wsdl lo hacemos de la siguiente manera: hacemos clic con el botón derecho del ratón sobre el nuevo proyecto creado y elegimos *New/Bpel process*:

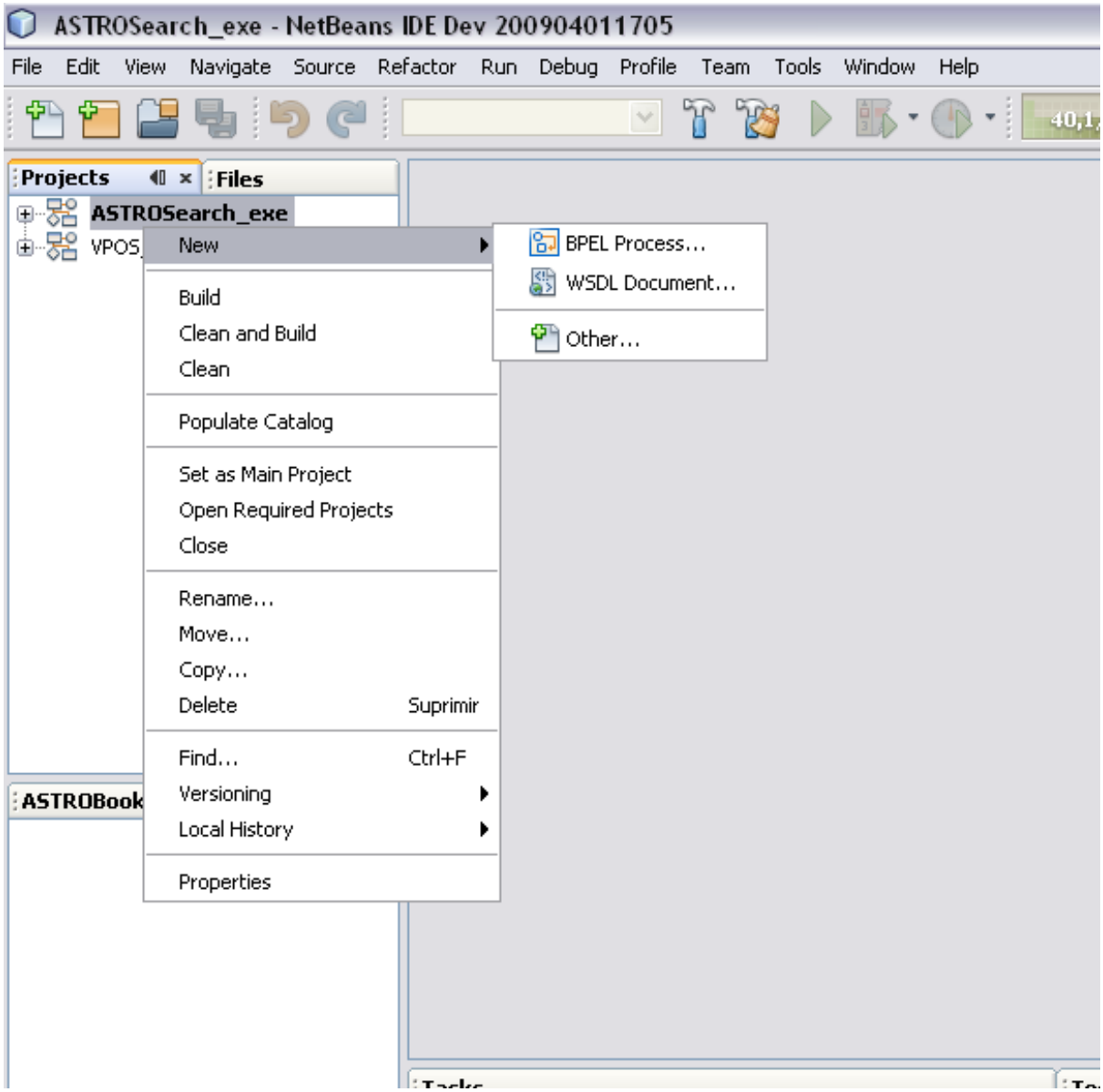


Figura 8.2: Nuevo archivo .bpel

Para añadir un nuevo fichero XML Schema `(.xsd)`, tenemos que hacerlo de una manera diferente. Debemos proceder así: Cliqueamos botón derecho encima del proyecto y seleccionamos *New/Other...* Ahora debemos seleccionar el tipo XML, y dentro seleccionamos el tipo XML Schema:

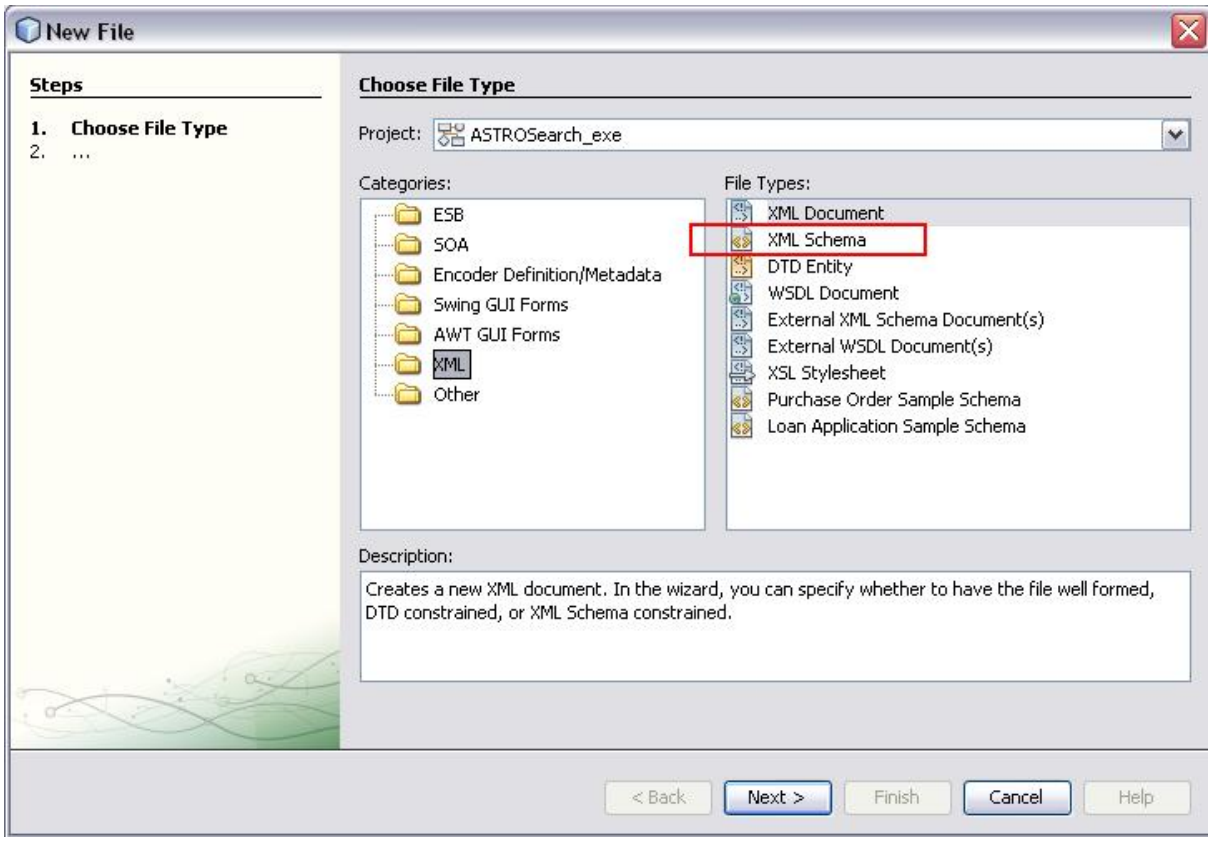


Figura 8.3: Nuevo archivo XML Schema con NetBeans

8.1.2. Uso de los ficheros XML Schema y WSDL

Para poder hacer el desplegado de nuestro proyecto, debemos incluirle a nuestro fichero BPEL los ficheros WSDL e XML Schema (si lo hubiera), para poder relacionar los mensajes con su interfaz. Abrimos el fichero .bpel. Nos encontraremos justo arriba del fichero una serie de pestañas, con los siguientes escritos: *Source, Design, Mapper, Logging*. Elegimos la pestaña “Design” y nos saldrá un esquema de lo que sería nuestro fichero .bpel:

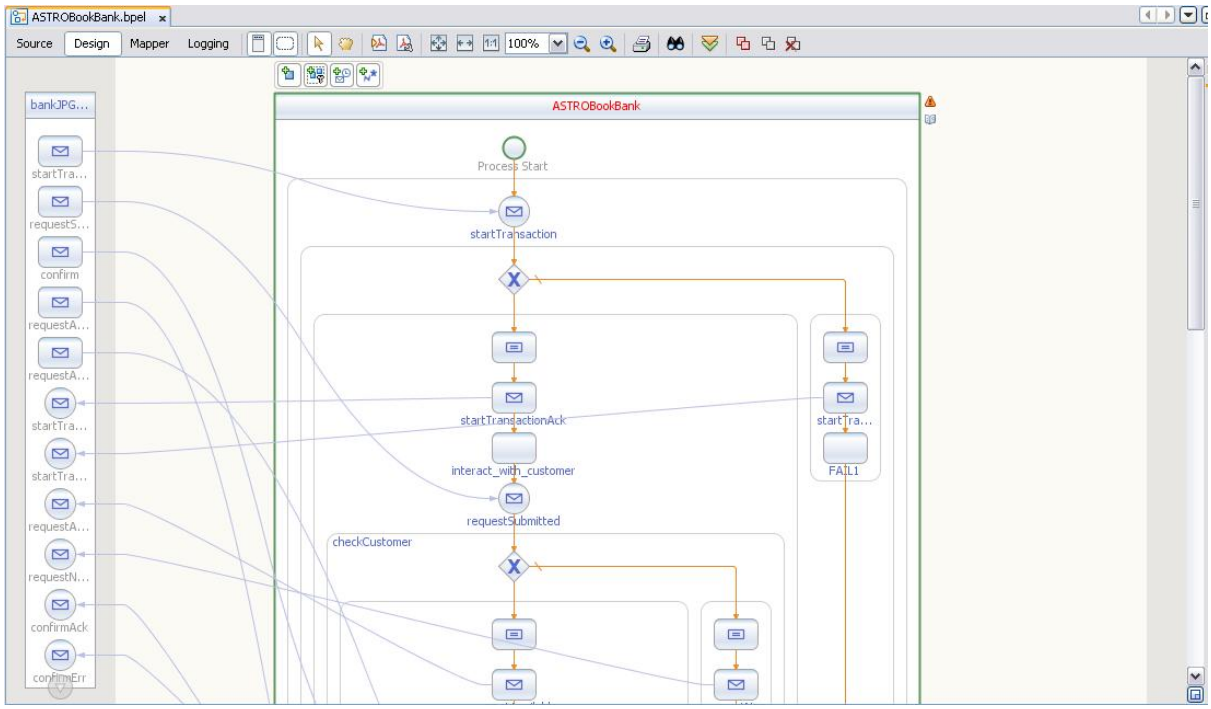


Figura 8.4: Esquema de un proceso BPEL

Sobre este esquema pulsamos botón derecho del ratón y nos saldrá un desplegable y elegiremos la opción “WSDL Import” o “Schema Import”, según el fichero que queramos incluir en el fichero .bpel:

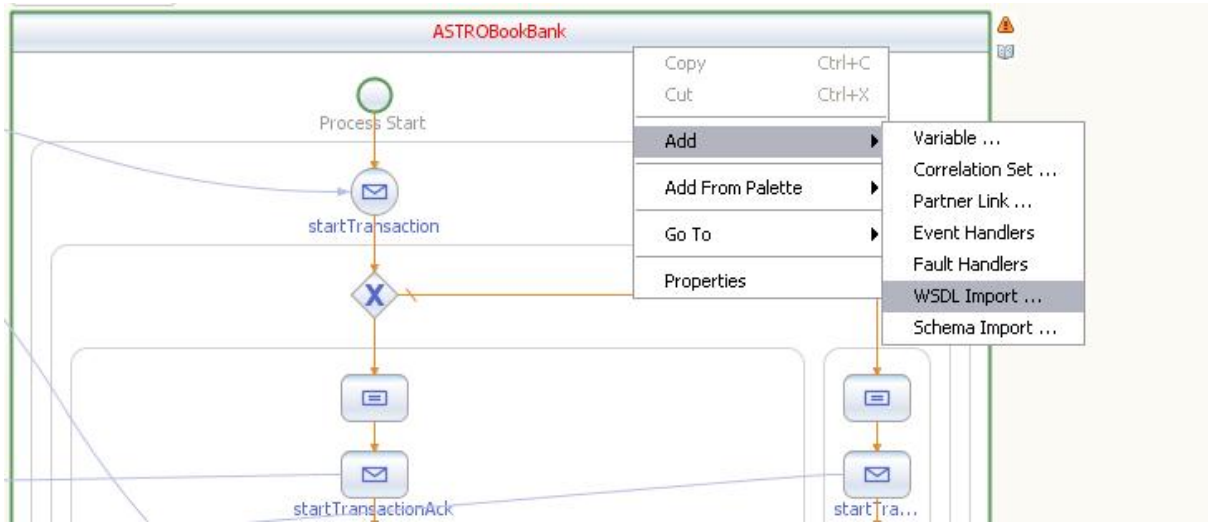


Figura 8.5: Exportar un archivo WSDL

Si seleccionamos la pestaña *Source* accederemos al código bpel de nuestra composición.

8.1.3. Desarrollo de composiciones con NetBeans

Para desplegar un proyecto BPEL con NetBeans, solo nos sirve para saber posibles errores a nivel de código, nunca las salidas que producirán nuestras composiciones. Para hacer un desplegado, cliqueamos botón derecho sobre el proyecto que queramos y seleccionar la opción “Build”:

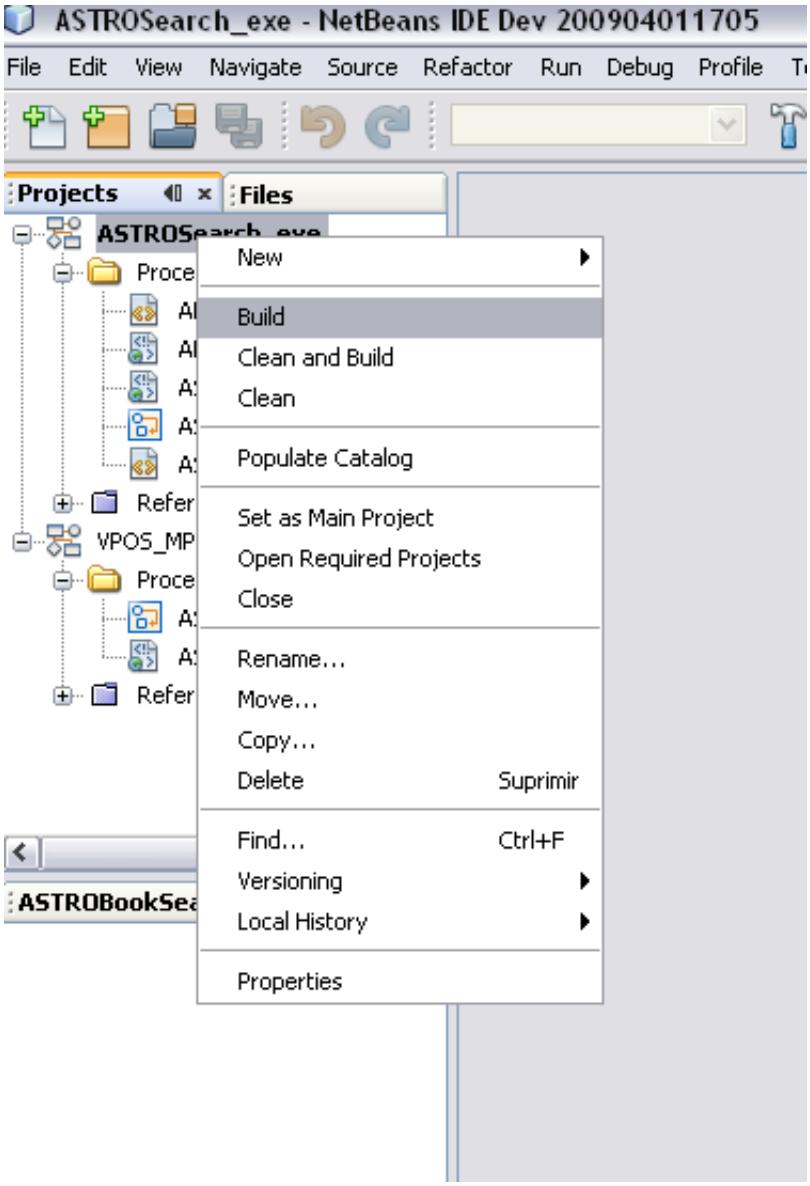
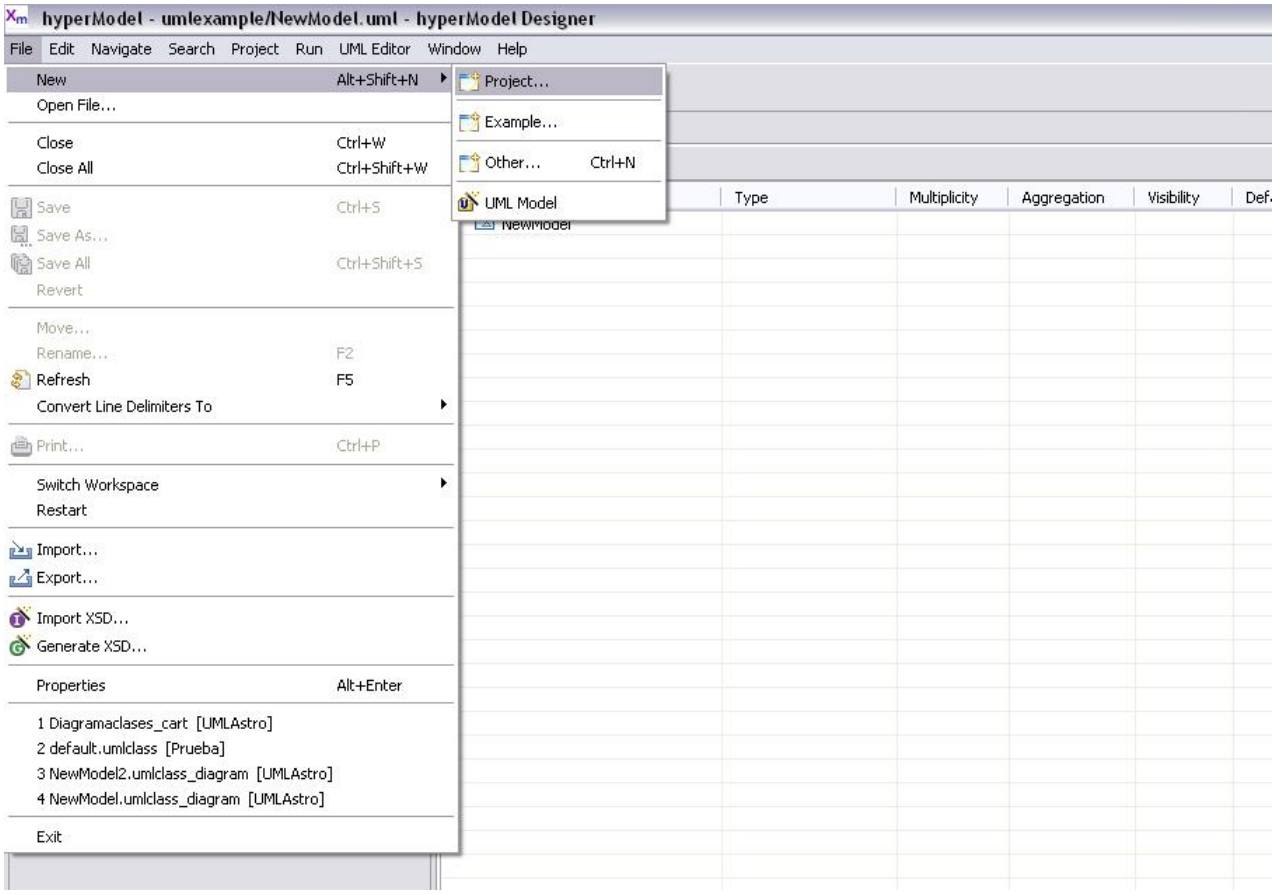


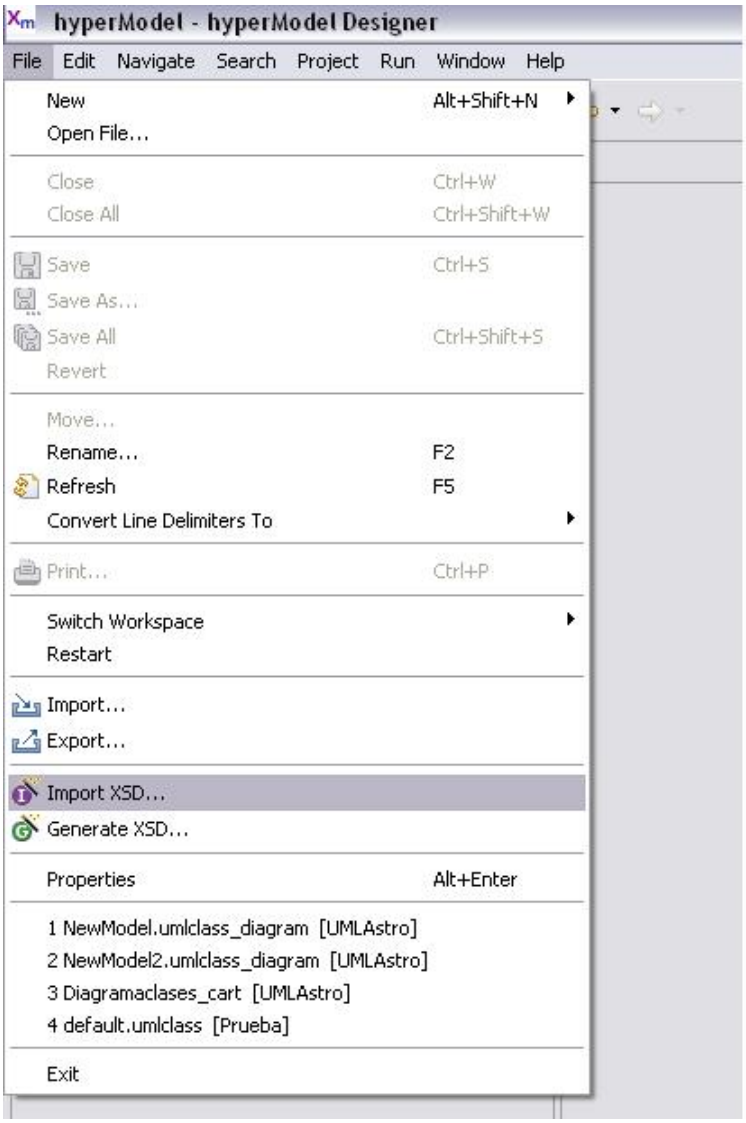
Figura 8.6: Desplegando un proyecto BPEL

8.2. Eclipse + HyperModel

HyperModel es un plugin de Eclipse para poder realizar diagramas de clase. Para realizar la parte de análisis, se necesitan hacer de las composiciones, sus correspondientes diagramas de clase. Para ello, Eclipse aporta un plugin que facilita las cosas. Lo que nos permite esta herramienta, es exportar un fichero XML Schema (.xsd) a UML. Para ello, primeramente creamos un nuevo proyecto de la siguiente manera: File/New/Project:



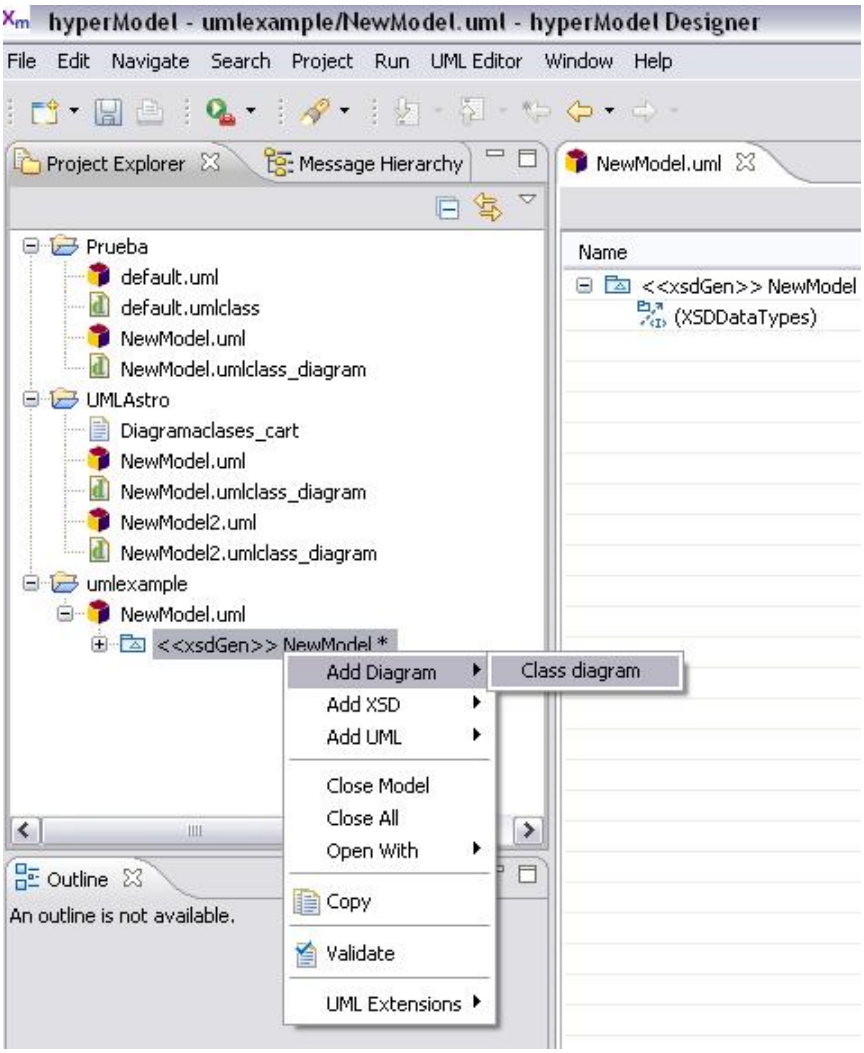
Con esto tenemos un nuevo proyecto creado para alojar en él los diagramas de clases que queramos. El siguiente paso es exportar el/los archivos .xsd que queramos pasar a uml. Para ello, hacemos clic en la barra de herramientas principal “File/Import XSD:



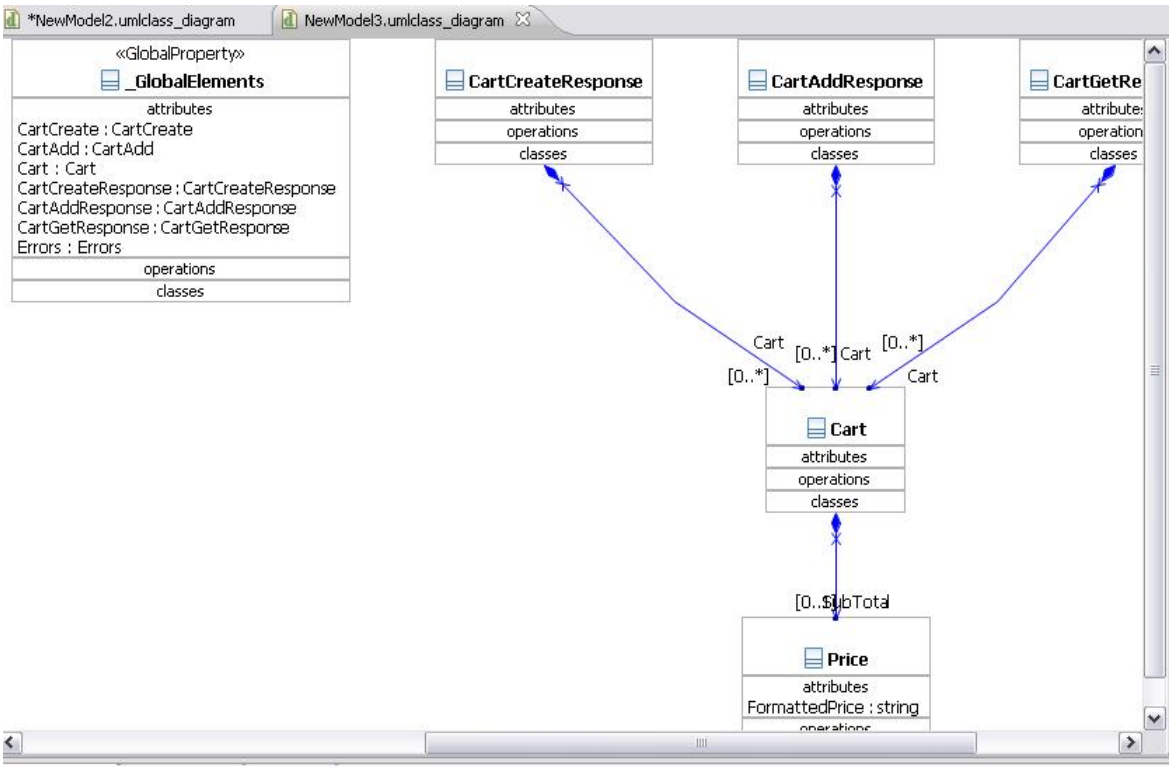
Nos saldrá una ventana en donde se nos pide que seleccionemos los archivos .xsd:

Seleccionamos los archivos y acto seguido tenemos que seleccionar un *Root Package*, que especifica qué carpeta se asigna al paquete raíz del modelo UML. Grandes conjuntos de archivos XSD se suelen organizar en una jerarquía de carpetas de archivos, y por lo general es conveniente para reflejar que la jerarquía misma carpeta en los paquetes de modelado UML. Para ello, deberá indicar al asistente que la carpeta de archivo debe ser la raíz de la jerarquía de paquetes UML (normalmente la carpeta de archivos más específico que contiene todos los esquemas y las carpetas a importar). Nos saldrá una ventana en donde se nos pide que seleccionemos los archivos .xsd:

Le damos a siguiente, y seleccionamos el proyecto que creamos antes. Le damos a "Next >" en las siguientes ventanas hasta finalizar el asistente. Con esto ya tenemos importado los archivos .xsd, pero ahora tenemos que pasarlo a UML. Sobre el archivo importado, hacemos clic derecho con el ratón y seleccionamos "Add diagram/ Class Diagram":



Nos volverá a salir un asistente, le ponemos un nombre al archivo y le damos a Finalizar. Ya tendremos el archivo XML Schema pasado a diagrama de clases:



8.3. mutationtool: herramienta desarrollada por el Grupo de Investigación

Las funciones de esta herramienta creada por el Grupo son las siguientes:

```
analyze bpel
apply bpel operator operand attribute
applyall bpel
run bpts bpel
compare bpts bpel xml bpel1...
comparefull bpts bpel xml bpel1...
compareout xml1 xml2
normalize bpel
```

Veamos algunas de ellas, las usadas para este Proyecto:

8.3.1. Arrancar motor ActiveBPEL

En primer lugar, tenemos que arrancar el motor de ActiveBPEL. Antes habría que ir a */bin/*, y dentro de este directorio escribir la siguiente orden en la consola:

```
$ ActiveBPEL.sh start
```

Pero ahora, tras unos cambios por parte de los miembros de grupo de investigación, puede hacerse desde cualquier directorio.

8.3.2. Ejecutando un proceso en BPEL

Para ejecutar nuestros procesos en bpel, debemos ir al directorio donde tengamos nuestra composición y abrir la consola. Escribimos la opción para ejecutarlo, que en cuestión es:

```
$ mutationtool run fichero.bpts fichero.bpel
```

Para comprobar los resultados de la ejecución, tenemos varias alternativas y todas son buenas para corroborar que efectivamente no hay errores.

8.3.3. Listar operadores de mutación usados

Con la siguiente orden podremos listar los operadores de mutación que nuestra composición está usando:

```
mutationtool analyze ficherobpel
```

Nota: Si queremos que en vez de mostrarlo por consola, nos lo muestre en un fichero, solo tenemos que volcarlo con el operador de redirección a un fichero > operadoresUtilizados.txt.

8.3.4. Generar mutantes de una composición BPEL

Para generar los mutantes resultantes de nuestra composición, solo tenemos que ejecutar la siguiente orden:

```
mutationtool applyall ficherobpel
```

Se encarga de generar los mutantes. El número de mutantes totales se corresponde con la suma del número de mutantes que hay para cada operador. El número de mutantes para cada operador se obtiene al multiplicar el número de veces que se puede aplicar cada operador de mutación por el número de atributos que tenga ese operador.

8.3.5. Obtener matrices de comparación

Para obtener las matrices en donde ver si tenemos mutantes vivos o muertos, ejecutamos la siguiente orden:

```
1 mutationtool compare ficherobpts ficherobpel ficherosalida.xml listamutantes
```

Nota: para poner todos los ficheros .bpel mutantes para generar la matriz resultante de ese operador en concreto, ponemos `m01*.bpel` y así podremos generar la matriz correspondiente al primer operador de mutación.

8.3.6. Normalizar una composición BPEL

Para identar el código BPEL, usamos la siguiente orden:

```
1 mutationtool normalize ficherobpel
```

Nota: para que no muestre el resultado de identar por la consola, debemos volcarlo en otro fichero.

8.3.7. Comprobación de *TestCases* con XMLEye

XMLEye [1] permite revisar qué casos de prueba fallan, qué casos de prueba pasan, y porqué falla:

[1] Aplicación creada por Antonio García Dominguez, donde se habló más a fondo en el Desarrollo del Proyecto.

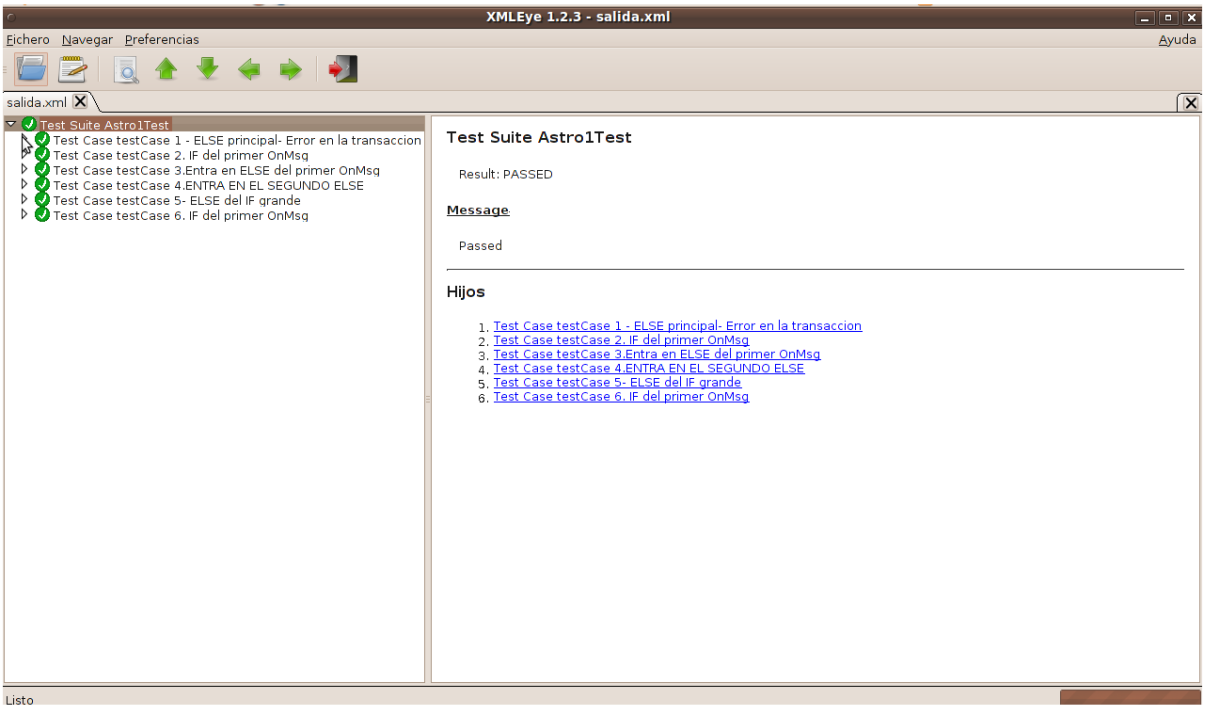


Figura 8.7: XMLEye

Podemos comprobar que las aspas rojas significan que ese caso de prueba ha fallado, pero para ver el fallo más detenidamente, solo hacemos clic y nos describirá el fallo:

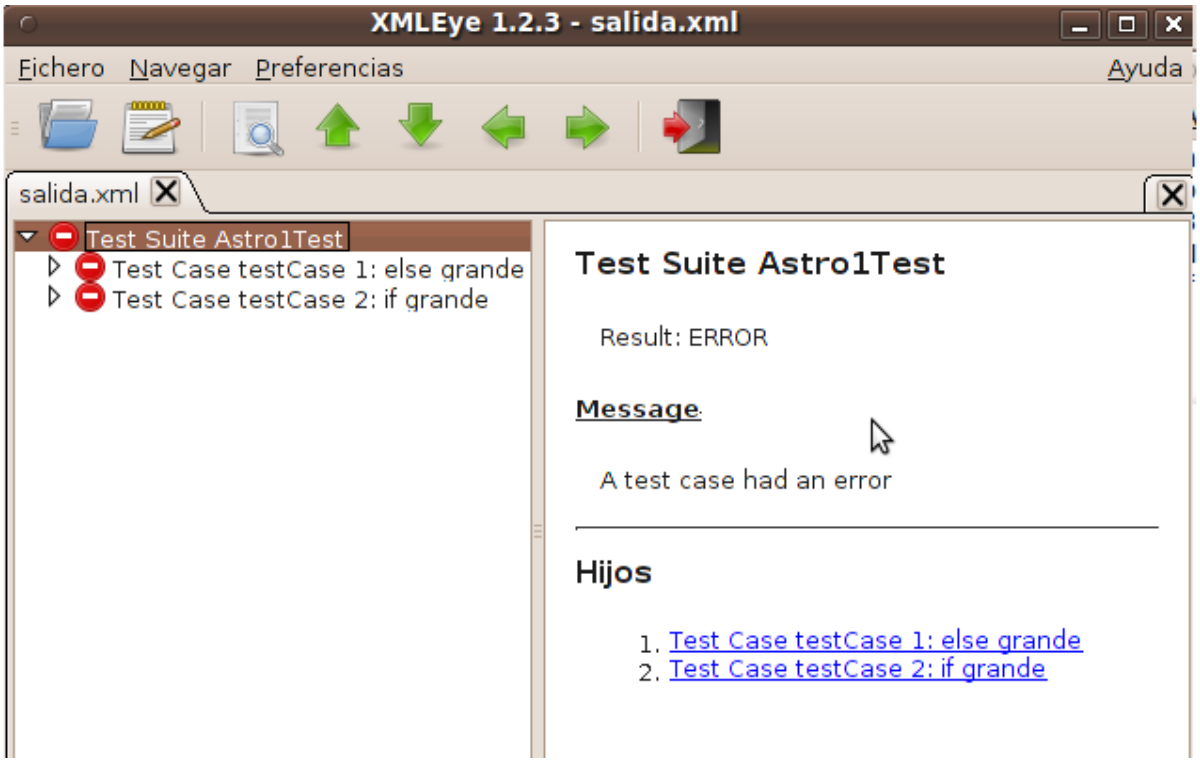


Figura 8.8: XMLEye

8.3.8. Comprobación de casos de prueba mediante registros

Mediante los registros que se generan al ejecutar procesos BPEL, podremos saber que en nuestra composición las actividades que componen nuestro BPEL se han realizado correctamente. Mediante estos ficheros también podremos saber si tenemos algún error. Entrando en el siguiente directorio encontraremos los ficheros de registro:

```
$ ~/AeBpelEngine/process-logs
```

8.3.9. Comprobación de código mediante ActiveBPEL

Mediante el desplegado, podemos saber si nuestra composición tiene errores en el código, en el siguiente enlace, lo podemos comprobar:

```
http://localhost:8080/BpelAdmin/
```

Le damos a *Deployment Log* y en un cuadro en la derecha vemos si todo ha salido bien nos saldrá un mensaje al final, donde ponga lo siguiente:

```
06/03/2010 01:14:35:746 [Astro1.bpr] [process.pdd] Succesfuly deployed
```

Si tuviéramos algún tipo de error, nos diría cuál es el error y dónde se produce y por qué. Como por ejemplo, las siguientes líneas nos avisan de un error, una actividad <assing>:

06/03/2010 01:14:35:715 [Astro1.bpr] [process.pdd] An element 'assign' from the BPEL namespace was added as an extensibility element since it appeared in an invalid location within the process.: in process.pdd

El registro de despliegue también están en:

```
~/AeBpelEngine/deployment-logs
```

8.3.10. Parar motor ActiveBPEL

Para para el motor, tan solo tenemos que ejecutar la orden:

```
1 $ ActiveBPEL.sh stop
```

O si sólo queremos reiniciarlo, debemos introducir la orden:

```
1 $ ActiveBPEL.sh restart
```

A Lenguaje WS-BPEL

A.1. Introducción

Lenguaje WS-BPEL [8] es un lenguaje basado en XML que permite especificar el comportamiento de un proceso de negocio basado en interacciones con Servicios Web. La estructura de un proceso WS-BPEL se divide en cuatro secciones: definición de relaciones con los socios externos, que son el cliente que utiliza el proceso de negocio y los WS a los que llama el proceso, definición de las variables que emplea el proceso, definición de los distintos tipos de manejadores que puede utilizar el proceso: manejadores de fallos y eventos, descripción del comportamiento del proceso de negocio; esto se logra a través de las actividades que proporciona el lenguaje.
Todos los elementos definidos anteriormente son globales si se declaran dentro del proceso. También existe la posibilidad de declararlos de forma local mediante el contenedor `scope`, que permite dividir el proceso de negocio en diferentes ámbitos.
Los principales elementos constructivos de un proceso WS-BPEL son las actividades, que pueden ser de dos tipos: básicas y estructuradas. Las básicas son las que realizan una determinada labor, mientras que las estructuradas pueden contener otras actividades y definen la lógica de negocio. A las actividades pueden asociarse un conjunto de atributos y de contenedores. Estos últimos pueden incluir diferentes elementos, que a su vez pueden tener atributos asociados. Además, WS-BPEL permite realizar actividades en paralelo y de forma sincronizada.

WS-BPEL es un acrónimo de Web Services Business Process Execution Language. WS-BPEL 2.0 es una versión actual, ya que en su anterior versión se llamaba BPEL4WS 1.1 (BPEL para Servicios Web). El objetivo de los Servicios Web es el esfuerzo para lograr la interoperabilidad entre aplicaciones utilizando los estándares Web. Los Servicios Web utilizan un modelo de integración ligeramente acoplados para permitir la integración flexible de sistemas heterogéneos en una serie de ámbitos tales como las empresas y consumidores, los negocios de comercio e integración de aplicaciones empresariales.
Las siguientes especificaciones básicas definió originalmente el espacio de Servicios Web: SOAP, Web Services Description Language (WSDL), y Universal Description, Discovery and Integration (UDDI). WSDL introduce una gramática común para los servicios con una descripción.
Los siguientes conceptos describen a los diferentes tipos de procesos de negocio:

- Los procesos de negocio son dependientes de la estructura de los datos.

- La capacidad de especificar las condiciones excepcionales y sus consecuencias, incluidas las secuencias de recuperación.
- Larga duración de interacciones, incluida las múltiples, a menudo se anidan las unidades de trabajo, cada una con sus requisitos propios. Los procesos de negocio con frecuencia requieren la coordinación socio cruzada de los resultados (éxito o fracaso) de las unidades de trabajo en los distintos niveles.

Estos conceptos básicos de WS-BPEL pueden ser aplicados en los dos tipos de composición: abstractas o ejecutables.
Un proceso abstracto, es un proceso especificado parcialmente, el cual no va a ser ejecutado y debe ser declarado como *abstracto*. Considerando que los procesos ejecutables están completamente especificados y por lo tanto puede ejecutarse, un proceso abstracto puede ocultar algunos de los detalles operacionales requeridos concretos.
WS-BPEL utiliza varias especificaciones XML: WSDL 1.1, XML Schema 1.0, XPath 1.0 y XSLT 1.0. WSDL es la interfaz del proceso WS-BPEL. XPath y XSLT proporcionan apoyo para la manipulación de datos. Todos los recursos externos y los socios están representados como los servicios de WSDL. Un proceso en BPEL especifica el orden exacto en el que los servicios Web participantes tienen que ser llamados. Pueden ser invocados en paralelo o secuencialmente.
También se pueden expresar comportamientos condicionados. Una invocación a un Servicio Web podría depender del resultado de una invocación anterior a otro servicio.
El proceso de negocio en BPEL recibe una petición, y para atender a esa petición invoca a los servicios web apropiados y responde al emisor original de la petición.
Dado que un proceso en BPEL se comunica con otros servicios web, se basará en sus descripciones WSDL para construir adecuadamente el repertorio de llamadas, es una extensión que WS-BPEL 2.0 hace a WSDL.

A.2. Estructura de un proceso de negocio WS-BPEL

Esta sección trata un breve resumen de lo que sería un proceso en BPEL. Proporciona una breve descripción, el resto de los detalles del lenguaje se describen a lo largo de este apéndice.
Un proceso en BPEL es un espacio, un contenedor donde se pueden declarar relaciones entre socios externos, manejadores para varios propósitos y lo más importante, las actividades que van a ser ejecutadas. El proceso BPEL tiene un par de atributos importantes, como son el nombre del proceso y las declaraciones de espacio de nombres (como se muestra en el ejemplo de abajo):

```
1 <process name="PrimerProcess"   <---- Nombre del proceso
2    targetNamespace="http://oasis-open.org/WSBPEL/Primer/"   <----
        Espacio de nombres
```

```
3     xmlns="http://docs.oasis-open.org/wsbpel/2.0/process/executable"
         />
```

Un proceso en BPEL está formado por actividades `<activities>`[1], y soporta actividades de varios tipos:

- Primitivas, construcciones básicas para tareas comunes como serían :
  - Invocar a un Servicio Web `<invoke>`
  - Esperar a que el cliente invoque al proceso de negocio `<receive>`
  - Generar una respuesta para una petición síncrona `<reply>`
  - Manipulación de valores de variables `<assign>`
  - Indicación de excepciones `<throw>`
  - Esperar un tiempo `<wait>`
  - Terminar un proceso `<terminate>`
- Estructuradas, que permiten combinar otras actividades:
  - Invocación secuencial de actividades `<sequence>`
  - Invocación en paralelo de actividades `<flow>`
  - Construcción iterativa `<while>`
  - Construcción if-else
  - Esperar la ocurrencia de algún evento de entre un conjunto de ellos `<pick>`

Un proceso en BPEL, además debe contener agentes externos `<partnerLinks>`, que son las referencias a servicios externos que tienen que ser invocados por el proceso BPEL. Son las diferentes partes que interactúan con los procesos de negocio en la invocación de una solicitud. Cada `<partnerLink>` se caracteriza por tener un `partnerLinkType` y uno o dos nombres de funciones. Esta información identifica la funcionalidad que debe tener proceso de negocio y el compañero para que el intercambio de mensajes tenga éxito.
También se puede hacer uso de variables locales o ámbito utilizando `<scope>`.

- La actividad `<receive>` permite al proceso de negocio esperar a un mensaje entrante. Esta actividad se completa cuando el mensaje llega. El atributo `<portType>` de esta actividad es opcional. Se recomienda incluirlo siempre para mayor claridad. En el caso de que estuviera incluido, el valor del atributo `<portType>` debe coincidir con el valor `<portType>` implicado por la combinación de la especificación `partnerLink` y el atributo `role` especificado en la actividad.

[1] Sólo se detallarán las actividades que se han utilizado en este Proyecto Fin de Carrera, dada a la cantidad de existencia de actividades para WS-BPEL 2.0

- La actividad <invoke> le permite al proceso de negocio llamar a una operación en un solo sentido o petición-respuesta sobre un portType ofrecido por un compañero o *partner*. En el caso de petición-respuesta *request-response*, la actividad invoke se completa cuando la respuesta es recibida. Al igual que en la actividad <receive>, el atributo <portType> de esta actividad es opcional. Se recomienda incluirlo siempre para mayor claridad. En el caso de que estuviera incluido, el valor del atributo <portType> debe coincidir con el valor <portType> implicado por la combinación de la especificación partnerLink y el atributo role especificado en la actividad.

- La actividad <assign> se usa para actualizar los valores de las variables con datos nuevos. Una actividad assign puede contener cualquier número de asignaciones, incluyendo <copy> para realizar la copia y asignación del nuevo valor que tomará la variable o las variables.

- La actividad <empty> es un "no-op" en un proceso de negocio. Esto es usual para la sincronización de las actividades concurrentes.

- La actividad <sequence> es usada para definir una colección de actividades para ser ejecutadas secuencialmente.

- La actividad <if> es usada para seleccionar exactamente una actividad para la ejecución desde un conjunto de opciones.

- La actividad <while> es usada para definir que una actividad hija va a ser repetida tantas veces como especifique el elemento anidado<condition>.

- La actividad <repeatUntil> es usada para definir que una actividad hija va a ser repetida hasta que la condición especificada <condition> llegue a ser verdadera. La diferencia con la actividad while, es que las actividades hijas deben ser ejecutadas al menos una vez.

- La actividad <pick> es usada para esperar para uno o varios posibles mensajes para ser recibidos. Cuando uno de estos mensajes llega, la actividad hija asociada es ejecutada. Cuando la actividad hija es completada entonces la actividad <pick> se completará. El atributo portType de una actividad <onMessage> es opcional, pero recomendable para mayor claridad.

- La actividad <flow> es usada para especificar una o más actividades para ser ejecutadas en paralelo.

A.2.1. Conectando los servicios

Un proceso WS-BPEL depende de un XML Schema y de WSDL para la definición de los tipos de datos y la interfaz del servicio. Las definiciones de procesos también se basan en otras construcciones, tales como partner link types, propiedades de variables (*variable properties*) y *property aliases*, las cuales están definidas dentro del lenguaje WSDL. El elemento `<import>` es usado en un proceso WS-BPEL para declarar una dependencia en un XML Schema externo o definiciones WSDL. Cualquier número de `<import>` pueden aparecer como hijos de un elemento `<process>`, y cada elemento tiene dos atributos opcionales y uno obligatorio:

- `namespace`. El atributo `namespace` especifica una cadena de texto con formato de dirección web que identifica las definiciones importadas. Este atributo es opcional.
- `location`. Este atributo contiene una cadena con formato web indicando la localización de un documento que contiene definiciones relevantes. Es lo que viene a ser el fichero `.wsdl`.
- `importType`. Este atributo es obligatorio, e identifica el tipo de documento que esta siendo importado. El valor del atributo debe ser: *http://www.w3.org/2001/XMLSchema*.

Características de agentes externos

Un `<partnerLinkType>` define la relación entre dos servicios mediante la especificación de roles proporcionados por cada servicio para recibir mensajes. Cada `<role>` especifica exactamente un `portType` en WSDL. El siguiente ejemplo muestra la sintaxis básica de una declaración `<partnerLinkType>`:

```
<plnk:partnerLinkType name="BuyerSellerLink">
   <plnk:role name="Buyer" portType="buy:BuyerPortType" />
   <plnk:role name="Seller" portType="sell:SellerPortType" />
</plnk:partnerLinkType>
```

PartnerLinks

Los servicios con el cual un proceso de negocio interactúa son modelados como agentes externos en WS-BPEL. Cada `<partnerLink>` es caracterizado por un `partnerLinkType`. Mas de un `<partnerLink>` puede ser caracterizado por el mismo `partnerLinkType`. Todos los `partnerLink` tiene un nombre (`name`), y cada `name` es usado para todas

interacciones con los servicios. Dentro de un agente externo, el rol del proceso de negocio es indicado por el atributo `myRole`, y el rol del compañero es indicado por el atributo `partnerRole`.

A.3. Propiedades de las variables

Los datos en un mensaje consiste conceptualmente en dos partes: aplicación de datos y protocolo relevante de datos.
Una definición de `<vprop: property>` crea un único nombre para un proceso en WS-BPEL, y está asociado con tipos XML Schema. Un típico uso de `<vprop:property>` en WS-BPEL es nombrar un token para una correlación o instancias de servicios con mensajes. Por ejemplo, el número de la seguridad social podría ser usado para identificar a un contribuyente en un proceso de negocio respecto a un asunto fiscal. Un número de seguridad social puede aparecer en diferentes tipos de mensajes, pero en el contexto de un proceso relacionado con los impuestos es especificado como un ID contribuyente. Por lo tanto un nombre es dado para el uso de este tipo definiendo un `<vprop:property>`, como en el siguiente ejemplo:

```
<wsdl:definitions name="properties"

   targetNamespace="http://example.com/properties.wsdl"
   xmlns:tns="http://example.com/properties.wsdl"
   xmlns:txtyp="http://example.com/taxTypes.xsd"
   xmlns:vprop="http://docs.oasis-open.org/wsbpel/2.0/varprop"
   xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">
   <!-- import schema taxTypes.xsd -->
   <!-- define a correlation property -->
   <vprop:property name="taxpayerNumber" type="txtyp:SSN" />

</wsdl:definitions>
```

El mecanismo de extensibilidad WSDL es usado para definir propiedades. La etiqueta de espacio de nombre y otros aspectos de uso de WSDL están disponibles para ello.
La sintaxis de la definición de una propiedad es un nuevo tipo de definición WSDL como sigue:

```
<wsdl:definitions name="NCName">
   <vprop:property name="NCName" type="QName"? element="QName"? />
</wsdl:definitions>
```

A.3.1. Definiendo Property Aliases

La noción de *alias* se introduce para asignar una propiedad a un campo de una parte del mensaje específico o valor de la variable. El nombre de la propiedad se convierte en un alias para la parte del mensaje y / o ubicación, y puede ser utilizado como tal en las expresiones y asignaciones. Como ejemplo, considere la siguiente definición WSDL mensaje:

```
<wsdl:definitions name="messages"

   targetNamespace="http://example.com/taxMessages.wsdl"
   xmlns:txtyp="http://example.com/taxTypes.xsd"
   xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">
   <!-- define a WSDL application message -->
   <wsdl:message name="taxpayerInfoMsg">
      <wsdl:part name="identification"
         element="txtyp:taxPayerInfoElem" />
   </wsdl:message>
</wsdl:definitions>
```

La definición de una propiedad y su ubicación en un campo concreto del mensaje se muestra en el siguiente fragmento de WSDL:

```
<wsdl:definitions name="properties"
   targetNamespace="http://example.com/properties.wsdl"
   xmlns:tns="http://example.com/properties.wsdl"
   xmlns:txtyp="http://example.com/taxTypes.xsd"
   xmlns:txmsg="http://example.com/taxMessages.wsdl" ...>
    <!-- define a correlation property -->
   <vprop:property name="taxpayerNumber" type="txtyp:SSN" />

   ...
  <vprop:propertyAlias propertyName="tns:taxpayerNumber"
      messageType="txmsg:taxpayerInfoMsg" part="identification">
      <vprop:query>txtyp:socialsecnumber</vprop:query>
   </vprop:propertyAlias>
   <vprop:propertyAlias propertyName="tns:taxpayerNumber"
      element="txtyp:taxPayerInfoElem">
      <vprop:query>txtyp:socialsecnumber</vprop:query>
   </vprop:propertyAlias>
</wsdl:definitions>
```

El primer `<vprop:propertyAlias>` define una propiedad nombrada `tns:taxpayerNumber` como un alias para una localización en la parte de la identificación de el tipo del mensaje `txmsg:taxpayerInfoMsg`.
El segundo `<vprop:propertyAlias>` proporciona una segunda definición para la

misma propiedad nombrada `tns:taxpayerNumber`, pero esta vez como un alias para una localización interna de el elemento `txtyp:taxPayerInfoElem`.
La sintaxis para una definición `<vprop:propertyAlias>` es:

```
<wsdl:definitions name="NCName" ...>
   <vprop:propertyAlias propertyName="QName"
      messageType="QName"?
      part="NCName"?
      type="QName"?
      element="QName"?>
      <vprop:query queryLanguage="anyURI"?>?
         queryContent
      </vprop:query>
   </vprop:propertyAlias>
   ...
</wsdl:definitions>
```

Un elemento `<vprop:propertyAlias>` debe usar una de estas tres combinaciones de atributos:

- `messageType` y `part`,
- `type` o
- `element`.

A.4. Correlación

La declaración de correlación se basa en las propiedades declarativas de los mensajes. Una propiedad es simplemente un "campo" en un mensaje identificado por una consulta. Esto sólo es posible cuando la estructura del mensaje está bien definido (por ejemplo, se describe mediante un XML Schema). Un conjunto de *tokens* de correlación es definido como un conjunto de propiedades compartidas por todos los mensajes en el grupo de correlación. Por lo tanto, un conjunto de propiedades es llamado conjunto de correlación *correlation set*.

```
<correlationSets>?
   <correlationSet name="NCName" properties="QName-list" />+
</correlationSet
```

Un `<correlationSet>` puede ser declarado dentro de un proceso o un *scope* de forma que es análoga a una declaración de variable. El nombre de un `<correlationSet>` debe ser único entre los nombres de todos los `<correlationSet>` definido dentro de los mismos. Este requisito se debe de cumplir de forma estática.

La correlación puede ser usada en todos las actividades donde intervienen paso de mensajes (<receive>, <reply>, <onMessage>, <onEvent>y <invoke>).
Las propiedades usadas en un <correlationSet> deben ser definidas usando tipos simples en un XML Schema. El correlation set es usado en actividades <invoke>, <receive> y <reply>. En las ramas de <onMessage> de las actividades <pick> también se usan los correlation set.
Estas especificaciones <correlation> identifican la correlación fija por su nombre y se utilizan para indicar que la correlación fija (es decir, los conjuntos de propiedades correspondientes) se producen en los mensajes enviados y recibidos. El inicio en una especificación de atributos <correlation> se emplea para indicar si el conjunto de correspondencias se está iniciando.
Después de que un conjunto de correlaciones es iniciado, los valores de las propiedades para un conjunto de correlaciones debe ser identificado para todos los mensajes en todas las operaciones que lleva el conjunto de correlación. Este restricción de correlación debe ser observada en todos los casos de iniciación de valores. El valor real del atributo initiate son: yes, join, no. El valor por defecto es no.

En el caso del <invoke>, cuando la operación invocada es una operación petición-respuesta (request-response), el atributo pattern dentro de un <correlation> es usado para indicar si la correlación se aplica a los mensajes salientes request, el mensaje de entrada response, o ambos request-response.
El atributo pattern usado en <invoke> es requerido para operaciones tipo request-response, y no permitidos cuando una operación en una sola dirección es llamada.

```
<correlations>
   <correlation set="NCName"
      initiate="yes|join|no"?
      pattern="request|response|request-response"? />+
</correlations>
```

A.5. Procesos Abstractos en WS-BPEL

Los procesos abstractos tienen múltiples casos de uso.
La base común es la forma sintáctica la cual todos los procesos abstractos WS-BPEL deben ajustarse. Las características sintácticas de la base común son:

- El atributo abstractProcessProfile debe existir si se quiere realizar un proceso abstracto.
- Todos los constructores de un proceso ejecutable son permitidos.
- Algunas construcciones de los procesos ejecutables pueden estar ocultas, de forma explícita mediante la inclusión de las extensiones del lenguaje opaco (*opaque language*).

- Un proceso abstracto debe cumplir con la restricciones de la sintaxis.
- Un proceso abstracto debe omitir la actividad `createInstance` (`<receive>` o `<pick>`) que es obligatorio para un proceso ejecutable WS-BPEL.

A.6. Lenguaje de consulta XPath

XPath fue diseñado para ser usado en XSLT. El lenguaje de ruta XML Path (XPath) es especialmente usado para seleccionar nodos o valores de un documento XML. En XPath un documento XML es visto como un árbol de nodos y una expresión XPath es una ruta abajo del árbol. Por ejemplo, la expresión:

```
1 /bpel:process/bpel:sequence/bpel:assign/bpel:copy/bpel:to/@part
```

selecciona el atributo `part` de el elemento a asignar en el último elemento `assign`.
XPath es el resultado de un esfuerzo por proporcionar una sintaxis y semántica comunes para funcionalidades compartidas entre Transformaciones XSL y XPointer. El objetivo principal de XPath es hacer frente a las partes de un documento XML. También proporciona servicios básicos para la manipulación de cadenas, números y booleanos.
XPath define una manera de calcular un valor de cadena para cada tipo de nodo. XPath es totalmente compatible con espacios de nombre XML.
La principal construcción sintáctica en XPath es la expresión. Una expresión se evalúa para obtener un objeto, que tiene uno de los siguientes cuatro tipos básicos:

- conjunto de nodos (colección desordenada de nodos sin duplicados)
- booleano
- números
- cadenas

Todo el procesamiento realizado con un fichero XML está basado en la posibilidad de direccionar o acceder a cada una de las partes que lo componen, de modo que podamos tratar cada uno de los elementos de forma diferenciada.
El tratamiento del fichero XML comienza por la localización del mismo a lo largo del conjunto de documentos existentes en el mundo. Para llevar a cabo esta localización de forma unívoca, se utilizan los *URI* (Unifom Resource Identifiers) cadenas de texto con formato web, de los cuales los *URL* (Unifom Resource Locators) son sin duda los más conocidos.
Una vez localizado el documento XML, la forma de seleccionar información dentro de él es mediante el uso de XPath, que es la abreviación de lo que se conoce como XML Path Language. Con XPath podremos seleccionar y hacer referencia a texto, elementos, atributos y cualquier otra información contenida dentro de un fichero XML.
XPath en sí es un lenguaje sofisticado y complejo, pero distinto de los lenguajes procedurales que solemos usar (C, C++, Basic, Java...). Además, como casi todo en el mundo

de XML, aún está en estado de desarrollo, por lo que no es fácil encontrar herramientas que incorporen todas sus funcionalidades.

A.7. WSDL

WSDL son las siglas de *Web Services Description Language*, un formato XML que se utilizara para describir Servicios Web. Describe la interfaz pública a los servicios WEB, la forma de comunicación, es decir, los requisitos del protocolo y los formatos de los mensajes necesarios para interactuar con los servicios.
WSDL se usa en combinación con SOAP y XML Schema.

Estructura del WSDL

Un documento wsdl tiene los siguientes elementos:

- Tipos de datos `<types>`: aquí se definen los tipos de datos usados en los mensajes. Se utilizan los tipos definidos en la especificación de XML Schema (por ejemplo `string`).
- Mensajes `<message>`: aquí se definen los elementos del mensaje. Cada mensaje puede tener varias partes. Las partes pueden ser de cualquiera de los tipos definidos en XML Schema o un tipo definido por nosotros mismos.
- Tipos de puerto `<portType>`: con este apartado definimos las operaciones permitidas y los mensajes intercambiados en el servicio.
- Bindings `<binding>`: aquí se especifican los protocolos de comunicación usados. El atributo `style` indica si la operación es RPC(mensajes que contienen parámetros y valores de retorno. Cuando el tipo de las partes de un mensaje es *element*, y es un tipo complejo definido por el programador) o un DOCUMENT (cuando el tipo de las partes de un mensaje es *type*, y es un tipo básico). Esta información puede utilizarse para seleccionar un modelo de programación apropiado. El valor de este atributo también afecta la manera en que se construye el cuerpo del mensaje SOAP.
- Servicios `<service>`: los servicios son un conjunto de puertos y dirección de los mismos.

A continuación, vemos un ejemplo de una interfaz WSDL de la composición ASTRO:

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <definitions name="ASTROBookBank" targetNamespace="http://j2ee.
     netbeans.org/wsdl/VPOS_MPS/ASTROBookBank"
3      xmlns="http://schemas.xmlsoap.org/wsdl/"
4      xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
```

```
5       xmlns:xsd="http://www.w3.org/2001/XMLSchema"
6       xmlns:tns="http://j2ee.netbeans.org/wsdl/VPOS_MPS/ASTROBookBank"
7       xmlns:plnk="http://docs.oasis-open.org/wsbpel/2.0/plnktype"
8       xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
9       xmlns:vprop="http://docs.oasis-open.org/wsbpel/2.0/varprop"
10      xmlns:bpws="http://schemas.xmlsoap.org/ws/2003/03/business-
           process/">
11      <types/>
12
13      <message name="startTransactionMsg">
14      <part name="shopDs" type="xsd:string"/>
15      <part name="shopId" type="xsd:string"/>
16      <part name="amount" type="xsd:string"/>
17      <part name="customerIP" type="xsd:string"/>
18      <part name="shopEmail" type="xsd:string"/>
19      <part name="key" type="xsd:string"/>
20  .....
21
22  <portType name="bankJPG_PT">
23      <operation name="startTransaction">
24        <input message="tns:startTransactionMsg"/>
25      </operation>
26  .....
27    </portType>
28
29  <plnk:partnerLinkType name="bankJPG_PLT">
30     <plnk:role name="bankJPG_Service" portType="tns:bankJPG_PT" />
31     <plnk:role name="bankJPG_Customer" portType="
          tns:bankJPG_CallbackPT"/>
32  </plnk:partnerLinkType>
33
34  <binding name="bankJPG_PLTServiceBinding" type="tns:bankJPG_PT">
35          <soap:binding style="document" transport="http://schemas.
              xmlsoap.org/soap/http"/>
36          <operation name="startTransaction">
37              <soap:operation style="document"/>
38              <input>
39                  <soap:body use="literal" namespace="http://j2ee.
                       netbeans.org/wsdl/VPOS_MPS/ASTROBookBank"/>
40              </input>
41          </operation>
42   ......
43
44    <service name="bankJPG_PLTService">
45          <port binding="tns:bankJPG_PLTServiceBinding" name="
              bankJPG_PLTServicePort">
```

```
46              <soap:address location="http://localhost:8080/active-
                   bpel/services/bankJPG_PLTService"/>
47          </port>
48      </service>
49  ......
50  <bpws:property name="key" type="xsd:string"/>
51  <bpws:propertyAlias messageType="tns:startTransactionMsg" part="key"
        propertyName="tns:key"/>
52  .....
```

A.8. SOAP

SOAP cuyas siglas significan *Simple Object Access Protocol* es un protocolo estándar que define como dos objetos en diferentes procesos pueden comunicarse por medio de intercambio de datos XML. Fue creado por Microsoft e IBM.

Las capacidades básicas de SOAP incluye lo siguiente:

- **Formato de mensajes estandarizados**. SOAP especifica cómo la información se empaqueta en un documento XML estandarizado (un mensaje SOAP) para ser trasladado en una comunicación.
- **Establecer correspondencias.** La especificación SOAP contiene un conjunto de convenciones para el mapeo de datos de aplicación en los mensajes SOAP, por ejemplo, para especificar una llamada a procedimiento remoto.
- **Modelo de procesamiento.** El modelo de procesamiento define las funciones de los remitentes y receptores de los mensajes SOAP, que especifica qué partes de un mensaje debe ser procesado por un rol.
- **Red de enlaces de protocolo.** SOAP especifica enlaces, que describen cómo los mensajes SOAP se van a transportar a través de HTTP, SMTP y otros protocolos de transporte.

El protocolo de comunicación SOAP es un solo sentido, sin estado, e intercambia información por medio de mensajes. Un mensaje SOAP es, en esencia, un documento XML estandarizado, que contiene algunos de los metadatos, e incorporando los datos de aplicación real. A pesar de la serie de convenios para el mapeo de los datos de aplicación para la transferencia de SOAP, no trata de interpretar estos datos o realizar cualquier operación semántica de otro tipo; los datos no es más que la carga útil del mensaje SOAP.

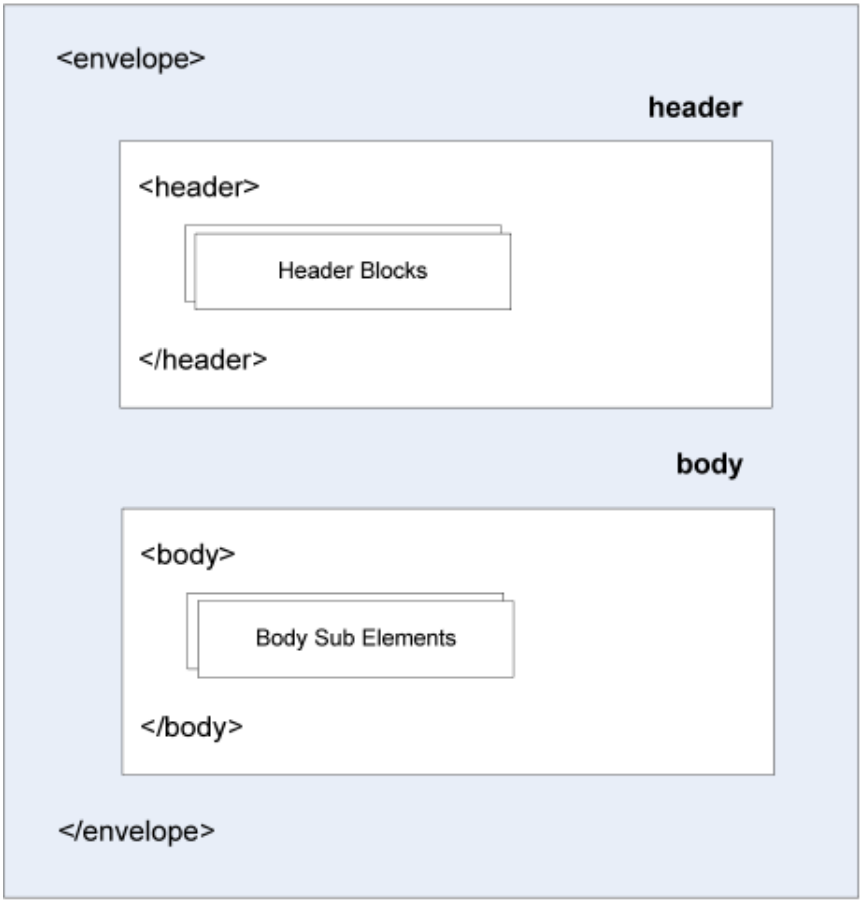


Figura A.1: Estructura de un mensaje SOAP

Procesamiento de mensajes SOAP

Como se puede observar en la figura A.1 de la página 108, un mensaje SOAP se compone de una envoltura SOAP, que incluye un encabezado SOAP y un cuerpo SOAP. El encabezado SOAP es opcional, es decir, se puede omitir. El cuerpo SOAP es obligatorio, es decir, cada mensaje SOAP válido debe contener un cuerpo. Un encabezado SOAP se compone de una serie de bloques de encabezado, que son los hijos de primer nivel de la cabecera. Del mismo modo, el cuerpo SOAP se compone de una serie de sub-elementos del cuerpo.

Los mensajes SOAP se transmiten entre nodos SOAP. Un nodo SOAP puede enviar y/o recibir mensajes. Hay un remitente inicial de un mensaje SOAP y un receptor final, y cualquier número (incluyendo el cero) de los intermediarios, los cuales reciben y envían mensajes.

A.9. Algunos cambios más relevantes de BPEL4WS 1.1 a WS-BPEL 2.0

Hay muchos cambios de una versión a otra, ya que WS-BPEL 2.0 no incluye todo lo que hay en BPEL4WS 1.1 y viceversa, por lo que no hay una correspondencia inmediata. A continuación vemos los cambios mas relevantes.

Cambios en actividades *assign*

Como vemos en el siguiente fragmento de código, si queremos copiar una cadena de texto a una variable vacía, en **BPEL4WS 1.1** se realiza del siguiente modo:

```
<assign>
          <copy>
             <from expression="'1JCNA3F1DE5FAWXC06G2'"/>  <--- vemos
                que si queremos copiar desde una cadena de texto,
                debemos poner el atributo ''expression''
             <to part="body" query="/ns1:CartCreate/
                ns1:AWSAccessKeyId" variable="CartCreateRequestMsg"/>
          </copy>

       </assign>
```

En **WS-BPEL 2.0** se realiza la copia de una cadena de texto a una variable del siguiente modo. Vemos como el atributo "*expression*" ha desaparecido y se deja simplemente con dobles comillas:

```
<copy>
         <from>"1JCNA3F1DE5FAWXC06G2"</from>
         <to part="body" variable="CartCreateRequestMsg">
           <query>/ns3:CartCreate/ns3:AWSAccessKeyId</query>
         </to>
       </copy>
```

Cambios en consultas *query*

Si queremos seleccionar un nodo dentro de una estructura de una variable definido en un XML Schema, en **BPEL4WS 1.1**, se define como un atributo dentro del elemento anidado `<to>` del siguiente modo:

```
<assign>
         <copy>
            <from expression="'1JCNA3F1DE5FAWXC06G2'"/>
            <to part="body" query="/ns1:CartCreate/
               ns1:AWSAccessKeyId" variable="CartCreateRequestMsg"/>
                 <--- Vemos que "query" forma parte del elemento <to
                 > como un atributo
         </copy>
      </assign>
```

En **WS-BPEL 2.0** se define como un elemento anidado, como sigue:

```
      <copy>
        <from>"1JCNA3F1DE5FAWXC06G2"</from>
        <to part="body" variable="CartCreateRequestMsg">
          <query>/ns3:CartCreate/ns3:AWSAccessKeyId</query> <---
             Vemos que es un elemento anidado
        </to>
      </copy>
```

Cambio en actividades condicionales

Ha habido cambios en actividades condicionales como es el uso de la actividad `switch` a `if`. En BPEL4WS 1.1 para evaluar una condición se hacía uso de la actividad condicional `switch` y su elemento anidado `<condition>` de la siguiente forma:

```
<switch>
         <case condition="bpws:getVariableData('
            CartCreateResponseMsg', 'body', '/ns1:CartCreateResponse
            /ns1:Cart/ns1:Request/ns1:IsValid')='True'&amp;quot;">
```

En WS-BPEL 2.0, para evaluar una condición se utiliza la actividad `<if>`, y se define de la siguiente manera:

```
 <if>
      <condition>$CartCreateResponseMsg.body, /
         ns3:CartCreateResponse/ns3:Cart/ns3:Request/ns3:IsValid='
         true'</condition>
      <sequence>
```

En el caso de tener que poner un caso contrario en una actividad que evalúe una condición, en BPEL4WS 1.1 se define así:

```
                                </case>
                                <otherwise>
```

Mientras que en WS-BPEL 2.0 se define de la siguiente forma:

```
                               <else>
```

Como podemos ver, se haría igual que en caso de cualquier otro lenguaje de programación estructurado como C, u orientado a objetos como C++ o Java.

Hay también un cambio bastante importante, cuando en una actividad `<invoke>`, contiene como elemento anidado un `<correlation>`, entonces el atributo de ese elemento anidado, para indicar que es una operación de salida, se expresa en BPEL4WS 1.1 de la siguiente forma:

```
<invoke inputVariable="loginAckMsg" name="loginAck" operation="
   loginAck" partnerLink="ASTROBookSearch_PLT" portType="
   ns3:ASTROBookSearch_CallbackPT">
<correlations>
       <correlation pattern="out" set="booksearch"/> <---
</correlations>
</invoke>
```

Vemos que el atributo *pattern* toma el valor *out* para indicar que es una operación de salida.
Sin embargo, en WS-BPEL 2.0, se indica de la siguiente forma:

```
<invoke inputVariable="searchResultMsg" name="searchResult"
   operation="itemSearchResponse" partnerLink="ASTROBookSearch_PLT"
   portType="ns0:ASTROBookSearch_CallbackPT">
<correlations>
        <correlation pattern="request" set="booksearch"/>  <---
</correlations>
</invoke>
```

Ahora el atributo *pattern* tomaría el valor *request /response /request-response*, dependiendo del tipo de operación.

B GAmera: un sistema para la generación automática de mutantes de composiciones WS-BPEL

Este apéndice trata de un sistema, denominado GAmera, para la generación automática de mutantes para composiciones WS-BPEL. Este sistema está basado en algoritmos genéticos e intenta minimizar el número de mutantes generados y ejecutados, independientemente del número y tipo de operadores de mutación.

B.1. Introducción.

El lenguaje WS-BPEL[1] [8] permite desarrollar procesos de negocios a partir de servicios Web (WS) preexistentes y mostrarlo como otro WS.
El el grupo de investigación está centrado en unas líneas de investigación orientadas a Servicios Web, ya que es necesario prestar especial atención a este lenguaje, pues que el impacto que están teniendo los Servicios Web en la actualidad está creciendo en el aspecto económico. El grupo ha publicado hasta la fecha diversos trabajos relacionados con la prueba de composiciones WS-BPEL, dedicados especialmente a la generación de casos de prueba, pero no la calidad de los casos de prueba.

SPI&FM cree que es interesante disponer de un sistema que nos permita medir la calidad de los distintos conjuntos de casos de prueba y mejorarla, para composiciones WS-BPEL. Una técnica adecuada es la generación de mutantes, que permite mejorar la calidad de un conjunto de casos de prueba mediante la generación y ejecución de mutantes. Los mutantes se generan mediante la aplicación de operadores de mutación al programa original.

Uno de los principales problemas que presenta GAmera es el alto coste computacional que supone la ejecución del gran número de mutantes que se genera a partir de cada composición.
Creemos que la aplicación de técnicas de optimización, tales como los algoritmos genéticos pueden ser de utilidad a la hora de generar los mejores mutantes, disminuyendo así el coste computacional de tener que ejecutar todos los mutantes.

[1] Apéndice A

El objetivo principal de este grupo de investigación es presentar el sistema de generación de mutantes para composiciones WS-BPEL: GAmera. Para la implementación de este sistema se han utilizado unos operadores de mutación definidos [2], donde se presentan 25 operadores. GAmera se compone de tres componentes: un analizador, un generador de mutantes basado en un algoritmo genético y un sistema que ejecuta y evalúa los mutantes.

B.2. Principales características de la prueba de mutaciones y algoritmos genéticos.

B.2.1. Prueba de mutaciones

La prueba de mutaciones es una técnica de prueba de caja blanca que introduce fallos simples en el programa a probar mediante la aplicación de operadores de mutación. Los programas resultantes se llaman *mutantes*. Los operadores de mutación son los errores más comunes que comenten los programadores al programar.
Por ejemplo, si tenemos un programa donde aparece la expresión & 10 >i , pues tenemos un operador de mutación relacional que reemplaza & por | y el operador >por el operador menor que <; si un caso de prueba es capaz de distinguir entre el programa original y el mutado, es decir, si las salidas son diferentes, decimos que el caso de prueba mata al mutante, y el mutante está *muerto*.
Por el contrario, si el caso de prueba no distingue entre el original y el mutado, decimos que el mutante sigue *vivo* para el conjunto de casos de prueba utilizado.
Una de las principales dificultades de aplicar la prueba de mutaciones es la existencia de mutantes equivalentes. Éstos presentan el mismo comportamiento que el programa original, es decir, la salida del mutante y del programa original es siempre la misma. Estos mutantes no deben confundirse con los mutantes resistentes (*stubborn nonequivalents mutants*), que se deben a que el conjunto de casos de prueba no tiene la suficiente calidad para poder detectarlos.
La calidad de los conjuntos de casos de prueba se mide mediante la puntuación de mutación (*mutation score*), que indica el número de mutantes muertos frente al número de mutantes no equivalentes. El conjunto de casos de prueba de mejor calidad es el que obtiene la mayor puntuación de mutación posible, es decir, consigue matar al mayor número de mutantes.
Otro de los problemas, es que la prueba de mutaciones tiene un alto coste computacional que supone la ejecución del gran número de mutantes que se suele generar a partir del programa a probar.
GAmera reduce el número de mutantes favoreciendo en la selección de mutantes de alta calidad. Los mutantes de alta calidad son los potencialmente equivalentes (mutan-

[2]Los operadores de mutación se definen B.3

tes equivalentes y resistentes) y a aquellos que son matados por un caso de prueba que sólo los mata a ellos (estos son mutantes difíciles de matar).

B.2.2. Algoritmos genéticos

Los algoritmos genéticos son una técnica de búsqueda probabilística basada en la teoría de la evolución y la selección natural. Constituye una estrategia efectiva de optimización heurística para funciones con muchos óptimos locales.
Los algoritmos genéticos trabajan con un conjunto de soluciones denominado *población*, formada a su vez por *individuos*. Los AG realizan un proceso de mejora de la población a través de las diferentes generaciones, por lo que resultan adecuados para la optimización. Lo más importante de los AG son su flexibilidad, simplicidad y habilidad para la depuración.
No existe un único tipo de AG, sino que existen varios tipos y familias de algoritmos que se diferencian en el esquema de codificación utilizado (binario, flotante,...). Cada individuo tendrá una aptitud que representa la calidad de la solución con respecto al problema a resolver. Los AG utilizan dos tipos de operadores: de selección y reproducción. Los de selección se encargan de seleccionar a los individuos de una población para su reproducción. Los operadores de reproducción, hay dos tipos: de cruce que generan dos tipos de individuos, llamados hijos. Los hijos heredan la información almacenada en los padres. Y los operadores de mutación (diferentes a los que se utilizan en la prueba de mutaciones) alternan la información almacenada en un individuo.

B.3. Operadores de mutación

GAmera introduce mutaciones en los procesos WS-BPEL aplicando los 25 operadores de mutación definidos dentro del grupo de investigación [15]. Estos operadores se clasifican en 4 categorías. Las categorías se diferencian por letras mayúsculas: I(mutación de identificadores), E (mutación de expresiones), A (mutación de actividades), X (mutaciones de condiciones excepcionalmente y eventos). Podemos ver la tabla de los operadores de mutación ordenadas por categorías en la tabla B.1.

B.4. Arquitectura de GAmera

GAmera está constituida por tres componentes: un analizador, un generador de mutantes basado en un algoritmo genético y un sistema que ejecuta y evalúa los mutantes[14].

Cuadro B.1: Operadores de mutación para WS-BPEL 2.0

OPERADOR	DESCRIPCIÓN
	MUTACIÓN DE IDENTIFICADORES
ISV	Sustituye el identificador de una variable por el de otra del mismo tipo
	MUTACIÓN DE EXPRESIONES
EAA	Sustituye un operador aritmético (+, −, *, `div`, `mod`) por otro del mismo tipo
EEU	Elimina el operador menos unario de cualquier expresión
ERR	Sustituye un operador relacional (<, >, >=, <=, =, !=) por otro del mismo tipo
ELL	Sustituye un operador lógico (`and`, `or`) por otro del mismo tipo
ECC	Sustituye un operador de camino (`/`, `//`) por otro del mismo tipo
ECN	Modifica una constante numérica incrementando o decrementando su valor en una unidad, añadiendo o eliminando un dígito
EMD	Modifica una expresión de duración cambiando por 0 o por la mitad el valor inicial
EMF	Modifica una expresión de fecha límite cambiando por 0 o por la mitad el valor inicial
	MUTACIÓN DE ACTIVIDADES
	Relacionados con la concurrencia
ACI	Cambia el atributo `createInstance` de las actividades de recepción de mensajes a *no*
AFP	Cambia una actividad `forEach` secuencial a paralela
ASF	Cambia una actividad `sequence` por una actividad `flow`
AIS	Cambia el atributo `isolated` de un `scope` a *no*
	No concurrentes
AIE	Elimina un elemento `elseif` o el elemento `else` de una actividad `if`
AWR	Cambia una actividad `while` por una actividad `repeatUntil` y viceversa
AJC	Elimina el atributo `joinCondition` de cualquier actividad en la que aparezca
ASI	Intercambia el orden de dos actividades hijas de una actividad `sequence`
APM	Elimina un elemento `onMessage` de una actividad `pick`
APA	Elimina el elemento `onAlarm` de una actividad `pick` o de un manejador de eventos
	MUTACIÓN DE CONDICIONES EXCEPCIONALES Y EVENTOS
XMF	Elimina un elemento `catch` o el elemento `catchAll` de un manejador de fallos
XMC	Elimina la definición de un manejador de compensación
XMT	Elimina la definición de un manejador de terminación
XTF	Cambia el fallo lanzado por una actividad `throw`
XER	Elimina una actividad `rethrow`
XEE	Elimina un elemento `onEvent` de un manejador de eventos

El analizador.

Es el primer componente de la herramienta que actúa. Recibe como entrada la composición WS-BPEL a probar y determina los operadores de mutación que se le pueden aplicar.

Generador de mutantes.

Los mutantes se generan a partir de la información que se obtiene a partir del analizador. La herramienta nos da la posibilidad de generar todos los mutantes posibles, o bien, un conjunto de estos que va a ser seleccionado por el algoritmo genético. En este último se llamará al componente denominado Generador genético de mutantes, que está compuesto por dos elementos. El primero, denominado Búsqueda Genética de Mutantes, es un algoritmo genético en el que cada individuo representa a un mutante, capaz de generar y seleccionar de forma automática un conjunto de mutantes. Esta selección se consigue aplicando una fórmula de aptitud que mide su calidad en función de si hay o no casos de prueba que lo matan[12]. El segundo elemento es el Conversor, que transforma un individuo de algoritmo genético en un mutante WS-BPEL. Para realizar esta conversión, se utilizan hojas de estilos XSLT, una por cada operador de mutación.

Ejecución de mutantes.

A medida que se van generando los mutantes, el sistema los ejecuta frente a un conjunto de casos de prueba, distinguiéndose tres posibles estados para cada mutante, según la salida que produce:

- **Muerto.** La salida del mutante es diferente a la del proceso original para al menos un caso de prueba.
- **Vivo.** La salida del mutante es la misma que la del proceso original para todos los casos de prueba.
- **Erróneo.** Se ha producido un error en el despliegue del mutante y no se ha podido ejecutar. La existencia de este estado permite determinar si el diseño o implementación de los operadores de mutación es adecuado, o bien, si se están generando mutantes que no se pueden desplegar.

Para la ejecución de los mutantes y el programa original, GAmera emplea el motor WS-BPEL 2.0, ActiveBPEL 4.1[1] y BPELUnit[11], una biblioteca de pruebas unitarias para WS-BPEL que utiliza ficheros XML para describir los casos de prueba. En la figura B.1 de la página 118, podemos ver la estructura que sigue GAmera.

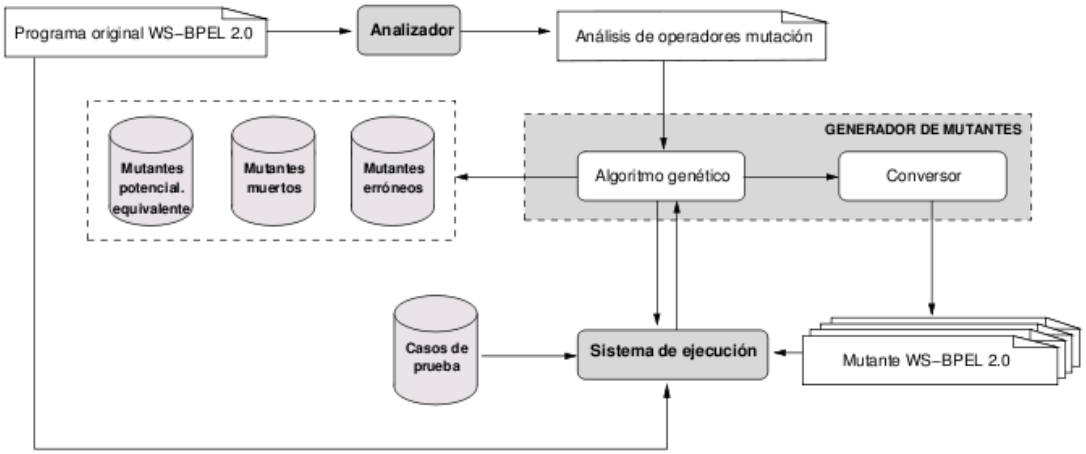


Figura B.1: Sistema de generación automática de mutantes para WS-BPEL

B.5. Funcionamiento de GAmera

Como sabemos por la sección anterior B.4, GAmera está compuesta por 3 componentes principales, describamos su funcionamiento paso a paso, componente a componente.

El analizador recibe de entrada la composición WS-BPEL 2.0 y genera la información necesaria para el generador de mutantes. El generador tiene como objetivo dos componentes: un algoritmo genético en el que cada individuo representa a un mutante, y un conversor de individuos a mutantes. Tiene como objetivo generar mutantes de calidad para las composiciones de WS. Por último, el sistema de ejecución se encarga de ejecutar el programa original y los mutantes frente a los casos de prueba y comparar las salidas que producen.

Los pasos para la ejecución del sistema generador de mutantes son los siguientes:

B.5.1. Paso 1: Análisis del proceso WS-BPEL

Es necesario identificar las diferentes elementos que sufrirán mutación. Esta tarea la realiza el analizador, que a partir de la definición del proceso WS-BPEL produce la lista de los operadores de mutación aplicables, así como el número de veces que se puede aplicar [3].

[3]Esta operación se puede realizar con *mutationtool analyze* como hemos visto ya en el manual de usuario

B.5.2. Paso 2: Creación del la población inicial

La salida del analizador es tomada por el algoritmo genético. La primera población del AG se genera aleatoriamente. El tamaño es uno de los parámetros de configuración del algoritmo.

La representación de un individuo sería de la siguiente manera:
(Operador, Localización, Atributo).

Operador:identifica al operador de mutación que se aplica. Es un valor entero entre 1 y 25 (que se corresponde con los operadores de mutación para el lenguaje WS-BPEL).
Localización:representa la localización dentro del programa original donde se aplicará el operador. También es un valor entero.
Atributo:representa la información necesaria para la aplicación del operador de mutación especificado en el campo *Operador*. Una mutación consiste en la alteración de un elemento del programa, este campo especifica cuál será el nuevo valor que tome el elemento. Los valores que pueda tomar también será un entero.

Una vez creada la población del algoritmo genético, para calcular la aptitud de cada individuo hay que convertir el individuo en el mutante que representa. El conversor recibe el individuo con los tres campos, y produce un fichero WS-BPEL del mutante asociado a dicha codificación. El mutante se obtiene a través de hojas de estilos XSLT a la composición WS-BPEL original.

B.5.3. Paso 3: ejecución de los mutantes

Para ejecutar los mutantes, primero se realiza el empaquetamiento que prepara al mutante para que pueda ser desplegado en el motor WS-BPEL 2.0. Luego, se realiza la ejecución en varias etapas: se despliega la composición WS-BPEL que representa al mutante, se invoca al mutante con cada uno de los casos de prueba que tengamos, se recogen los mensajes de respuesta y se finaliza el proceso WS-BPEL. A destacar por el grupo de investigación, el sistema permite llamar a Servicios Web reales o sustituirlos por *mockups*.
En el paso de comparación se compara la salida del mutante con la de la composición original para decidir si el mutante ha muerto o sigue vivo. Se realiza una comparación uno a uno de los mensajes de respuesta SOAP.
A partir de estas comparaciones se genera la matriz de ejecución: para cada fila, damos el valor 1 a la posición correspondiente al caso de prueba para el que la salida del mutante es diferente a la del proceso original. El resto de celdas se ponen a 0. En el caso de haber existido un error en el despliegue del mutante se da el valor 2 a toda la fila.
Así, una vez finalizada la ejecución de todos los mutantes, tendremos una matriz de

ejecución.
La figura B.2 de la página 121 muestra los diferentes pasos del sistema de ejecución.

B.5.4. Paso 4: Cálculo de la aptitud del individuo

En la matriz de ejecución que se crea con los resultados de los mutantes vivos o muertos, se establecen las siguientes relaciones:

- Un mutante puede ser matado por un caso de prueba o seguir vivo:
  $\sum_{j=1}^{T} m_i j \in [0,1], \forall i$
- Un caso de prueba puede que no mate a ninguno:
  $\sum_{i=1}^{M} m_i j \in [0,M], \forall j$
- Un mutante puede ser erróneo:
  $\sum_{j=1}^{T} m_i j = 2 \cdot T, \forall i$

Cuando un caso de prueba no mata a ningún mutante significa que: o el caso de prueba tiene baja calidad, o bien los mutantes generados no son adecuados para ese caso de prueba.

B.5.5. Paso 5: Generación de una nueva población

A partir de la segunda generación, cada población del algoritmo genético se construye en dos fases:

1. Generación aleatoria. Un parámetro de configuración del AG permite determinar el porcentaje de individuos de la población que se van a generar aleatoriamente.
2. Operaciones de cruce y mutación. La población se completa con la generación de nuevos individuos mediante la aplicación de operaciones de cruce y mutación a aquellos individuos seleccionados.

B.5.6. Paso 6: Comprobación del criterio de terminación

El algoritmo genético finalizará cuando se genere el porcentaje de mutantes indicado sobre el número máximo que permite la composición original. El porcentaje es un parámetro del algoritmo.

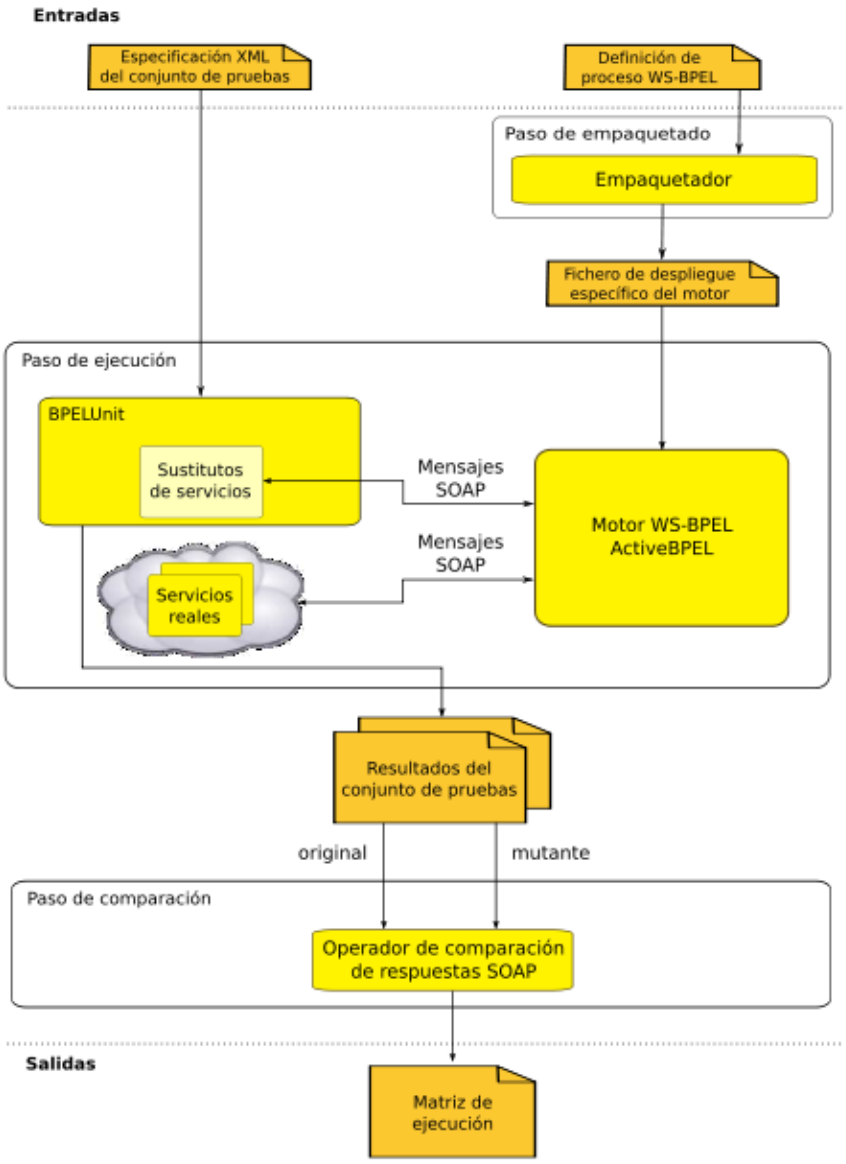


Figura B.2: El sistema de ejecución

C Glosario

Servicios Web

Permiten el intercambio de información en sistemas distribuidos heterogéneos. Forman una arquitectura que permite describir, publicar, descubrir, solicitar e invocar diferentes servicios empleando la infraestructura de internet.
El lenguaje WS-BPEL, especifica el comportamiento de un proceso de negocio basado en interacciones con Servicios Web.

Algoritmos genéticos

Son técnicas de búsqueda probabilística basadas en la teoría de la evolución y la selección natural.
Supervivencia de los mejores y carácter hereditario de las características. Se favorecen a los mejores y generan nuevos individuos (recombinación y mutación de información).

Análisis de mutaciones

Proceso de medir la calidad de conjuntos de casos de prueba.

Mutantes

Programas que contienen una única diferencia con respecto al programa original. Se generan aplicando al código fuente un conjunto de reglas definidas previamente, los operadores de mutación.

Operadores de mutación

Conjunto de reglas que introducen pequeños cambios sintácticos basados en los errores que suelen cometer habitualmente los programadores, o bien pretenden forzar ciertos criterios de cobertura del código.

Mutantes equivalentes

Mutantes que siempre provocan la misma salida que el programa original, por lo que no va a existir ningún caso de prueba que permita salida que el programa original.

Mutantes muertos

Mutantes que tiene una salida distinta a la del programa original, es decir, que existe un caso de prueba que permite que el programa original sea diferente al mutado, por lo que se dice que está muerto.

Mockup

Un *mockup* es un fichero con extensión `.bpts` (BPEL Test Suite), el cual se representan los mensajes enviados y recibidos con el cliente; en resumen, un conjunto de casos de prueba para un una composición BPEL determinada.

GNU Free Documentation License

Version 1.3, 3 November 2008

Copyright © 2000, 2001, 2002, 2007, 2008 Free Software Foundation, Inc.

<http://fsf.org/>

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

Preamble

The purpose of this License is to make a manual, textbook, or other functional and useful document "free" in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or non-commercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The "**Document**", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed

as "**you**". You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A "**Modified Version**" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A "**Secondary Section**" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The "**Invariant Sections**" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The "**Cover Texts**" are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A "**Transparent**" copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not "Transparent" is called "**Opaque**".

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The "**Title Page**" means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

The "**publisher**" means any person or entity that distributes copies of the Document to the public.

A section "**Entitled XYZ**" means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as "**Acknowledgements**", "**Dedications**", "**Endorsements**", or "**History**".) To "**Preserve the Title**" of such a section when you modify the Document means that it remains a section "Entitled XYZ" according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally

prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.

B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.

C. State on the Title page the name of the publisher of the Modified Version, as the publisher.

D. Preserve all the copyright notices of the Document.

E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.

F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.

G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.

H. Include an unaltered copy of this License.

I. Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.

J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.

K. For any section Entitled "Acknowledgements" or "Dedications", Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.

L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.

M. Delete any section Entitled "Endorsements". Such a section may not be included in the Modified Version.

N. Do not retitle any existing section to be Entitled "Endorsements" or to conflict in title with any Invariant Section.

O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section Entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties—for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified

Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled “History” in the various original documents, forming one section Entitled “History”; likewise combine any sections Entitled “Acknowledgements”, and any sections Entitled “Dedications”. You must delete all sections Entitled “Endorsements”.

6. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

7. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an "aggregate" if the copyright resulting from the compilation is not used to limit the legal rights of the compilation's users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document's Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

8. TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

If a section in the Document is Entitled "Acknowledgements", "Dedications", or "History", the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

9. TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense, or distribute it is void, and will automatically terminate your rights under this License.

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, receipt of a copy of some or all of the same material does not give you any rights to use it.

10. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See http://www.gnu.org/copyleft/.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation. If the Document specifies that a proxy can decide which future versions of this License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Document.

11. RELICENSING

"Massive Multiauthor Collaboration Site" (or "MMC Site") means any World Wide Web server that publishes copyrightable works and also provides prominent facilities for anybody to edit those works. A public wiki that anybody can edit is an example of such a server. A "Massive Multiauthor Collaboration" (or "MMC") contained in the site means any set of copyrightable works thus published on the MMC site.

"CC-BY-SA" means the Creative Commons Attribution-Share Alike 3.0 license published by Creative Commons Corporation, a not-for-profit corporation with a principal place of business in San Francisco, California, as well as future copyleft versions of that license published by that same organization.

"Incorporate" means to publish or republish a Document, in whole or in part, as part of another Document.

An MMC is "eligible for relicensing" if it is licensed under this License, and if all works that were first published under this License somewhere other than this MMC, and subsequently incorporated in whole or in part into the MMC, (1) had no cover texts or invariant sections, and (2) were thus incorporated prior to November 1, 2008.

The operator of an MMC Site may republish an MMC contained in the site under CC-BY-SA on the same site at any time before August 1, 2009, provided the MMC is eligible for relicensing.

Bibliografía

[1] ActiveVOS. Activebpel WS-BPEL and BPEL4WS engine. http://sourceforge.net/projects/activebpel.

[2] C. Michael Pilato Ben Collins-Sussman, Brian W. Fitzpatrick. *Version Control with Subversion*. Creative Commons, 2006.

[3] Cascales Salinas, Bernardo et al. *LaTeX, una imprenta en sus manos*. ADI, 2000.

[4] Robin Cover. XML Schema. http://www.w3.org/XML/Schema.html.

[5] Elena Orta Cuevas. *Apuntes de Ingeniería del Software I para la Ingeniería Técnica en Informática de Gestión*. Universidad de Cádiz, 2009.

[6] Juan José Dominguez-Jimenez, Antonia Estero-Botaro, Antonio Garcia-Dominguez, and Inmaculada Medina-Bulo. GAmera: an automatic mutant generation system for WS-BPEL compositions. In *Proceedings of the 7th IEEE European Conference on Web Services*, pages 97–106, Eindhoven, The Netherlands, November 2009.

[7] Manuel Palomo Duarte. Dossier de presentación de takuan. https://neptuno.uca.es/redmine/documents/18.

[8] IBM-OASIS. *Web Services Business Process Execution Language Version 2.0. OASIS Standard*. abril 2007.

[9] C. Larman. *UML y Patrones. Una introduccion al analisis y diseño orientado a objetos y al proceso unificado*. Ed. Prentice Hall, 2003.

[10] Joaquin Ataz Lopez. Creación de ficheros LaTeX con GNU Emacs. ftp://ftp.dante.de/tex-archive/info/spanish/guia-atx/guia-atx.pdf.

[11] Philip Mayer. *Design and Implementation of a Framework for Testing BPEL Compositions*. 2006.

[12] Dominguez Jimenez, Juan Jose Medina Bulo,Inmaculada, Estero Botaro, Antonia. Una arquitectura para la generación de casos de prueba de composiciones WS-BPEL basada en mutaciones. *JSWEB*, 2009.

[13] Dominguez Jimenez, Juan Jose Medina Bulo,Inmaculada, Estero Botaro, Antonia and Gutierrez Madroñal, Lorena. Gamera: Un sistema para la generación automática de mutantes de composiciones WS-BPEL. *JSWEB*, 2009.

[14] Dominguez Jimenez, Juan Jose Medina Bulo,Inmaculada, Estero Botaro, Antonia and Palomo Lozano, Francisco. Un sistema para la generación automática de mutantes de composiciones WS-BPEL. *JSWEB*, 2009.

[15] Estero Botaro, Antonia Medina Bulo,Inmaculada and Palomo Lozano, Francisco. Operadores de mutación para ws-bpel 2.0. *JSWEB*, 2009.

[16] Sun Microsystems. GlassFish Server Open Source Edition. https://glassfish.dev.java.net/.

[17] Bernardo Cascales Salinas. *El libro de LaTeX*. Pearson Educación, 2003.

[18] W3C. Simple Object Access Protocol. http://es.wikipedia.org/wiki/Simple_Object_Access_Protocol.

[19] W3C. Web Services Description Language (WSDL) 1.1. http://www.w3.org/TR/wsdl.

[20] Wikibooks. The Book of LaTeX. http://en.wikibooks.org/wiki/LaTeX.

[21] Frank Mittelbach y Michel Goossens. *The LaTeX Companion*. Addison-Wesley, 2004.

[22] James Clark y Steve DeRose (Inso Corp. and Brown University). XML Path Language (XPath). http://www.w3.org/TR/xpath/.